Amazon Simple Storage Service

API Reference

API Version 2006-03-01

Amazon Simple Storage Service: API Reference

Copyright © 2014 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

The following are trademarks of Amazon Web Services, Inc.: Amazon, Amazon Web Services Design, AWS, Amazon CloudFront, Cloudfront, Amazon DevPay, DynamoDB, ElastiCache, Amazon EC2, Amazon Elastic Compute Cloud, Amazon Glacier, Kindle, Kindle Fire, AWS Marketplace Design, Mechanical Turk, Amazon Redshift, Amazon Route 53, Amazon S3, Amazon VPC. In addition, Amazon.com graphics, logos, page headers, button icons, scripts, and service names are trademarks, or trade dress of Amazon in the U.S. and/or other countries. Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon.

All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

Welcome to Amazon S3 1
How Do I...? 1
Introduction 2
Error Responses 3
List of Error Codes 3
REST Error Responses 9
REST API 11
Common Request Headers 12
Common Response Headers 14
Authenticating Requests (AWS Signature Version 4) 15
Authentication Methods 16
Introduction to Signing Requests 16
Using an Authorization Header 17
Using Query Parameters 38
Examples: Signature Calculations 43
Authenticating HTTP POST Requests 45
Amazon S3 Signature Version 4 Authentication Specific Policy Keys 47
Browser-Based Uploads Using POST 51
Calculating a Signature 52
Creating HTML Forms 52
Creating a POST Policy 56
Upload Examples 61
Additional Considerations 64
Operations on the Service 65
GET Service 65
Operations on Buckets 68
DELETE Bucket 69
DELETE Bucket cors 71
DELETE Bucket lifecycle 73
DELETE Bucket policy 75
DELETE Bucket tagging 77
DELETE Bucket website 79
GET Bucket (List Objects) 81
GET Bucket acl 89
GET Bucket cors 92
GET Bucket lifecycle 95
GET Bucket policy 101
GET Bucket location 103
GET Bucket logging 105
GET Bucket notification 108
GET Bucket tagging 111
GET Bucket Object versions 114
GET Bucket requestPayment 126
GET Bucket versioning 128
GET Bucket website 131
HEAD Bucket 133
List Multipart Uploads 135
PUT Bucket 144
PUT Bucket acl 150
PUT Bucket cors 157
PUT Bucket lifecycle 162
PUT Bucket policy 171
PUT Bucket logging 173
PUT Bucket notification 178
PUT Bucket tagging 182

PUT Bucket requestPayment 185
PUT Bucket versioning 187
PUT Bucket website 191
Operations on Objects 198
DELETE Object 200
Delete Multiple Objects 203
GET Object 212
GET Object ACL 222
GET Object torrent 226
HEAD Object 228
OPTIONS object 235
POST Object 238
POST Object restore 247
PUT Object 250
PUT Object acl 262
PUT Object - Copy 269
Initiate Multipart Upload 282
Upload Part 290
Upload Part - Copy 295
Complete Multipart Upload 302
Abort Multipart Upload 308
List Parts 310
Resources 315
Document History 317
Appendix 323
Operations on the Service 323
ListAllMyBuckets 323
Operations on Buckets 325
CreateBucket 325
DeleteBucket 326
ListBucket 327
GetBucketAccessControlPolicy 330
SetBucketAccessControlPolicy 331
GetBucketLoggingStatus 332
SetBucketLoggingStatus 333
Operations on Objects 334
PutObjectInline 334
PutObject 336
CopyObject 338
GetObject 342
GetObjectExtended 347
DeleteObject 348
GetObjectAccessControlPolicy 348
SetObjectAccessControlPolicy 349
SOAP Error Responses 350
Glossary 352

Welcome to Amazon S3

This is the *Amazon Simple Storage Service API Reference*. It explains the Amazon Simple Storage Service (Amazon S3) application programming interface. It describes various API operations, related request and response structures, and error codes.

Amazon S3 is a web service that enables you to store data in the cloud. You can then download the data or use the data with other AWS services, such as Amazon Elastic Compute Cloud (Amazon EC2). For information about Amazon EC2, see Amazon Elastic Compute Cloud (Amazon EC2)).

How Do I...?

Information	Relevant Sections
General product overview and pricing	Amazon Simple Storage Service (Amazon S3)
List of REST operations	REST API (p. 11)
List of SOAP operations	Appendix: SOAP API (p. 323)
Amazon S3 error codes and descriptions	List of Error Codes (p. 3)

Amazon S3 API Reference Introduction

This application programming interface reference explains Amazon S3 operations, their parameters, responses, and errors. There are separate sections for the REST and SOAP APIs, which include example requests and responses.

The location of the latest Amazon S3 WSDL is http://doc.s3.amazonaws.com/2006-03-01/AmazonS3.wsdl.

Note
SOAP support over HTTP is deprecated, but it is still available over HTTPS. New Amazon S3 features will not be supported for SOAP. We recommend that you use either the REST API or the AWS SDKs

Error Responses

This section provides reference information about Amazon S3 errors.

Note
SOAP support over HTTP is deprecated, but it is still available over HTTPS. New Amazon S3 features will not be supported for SOAP. We recommend that you use either the REST API or the AWS SDKs.

Topics

- List of Error Codes (p. 3)
- REST Error Responses (p. 9)

List of Error Codes

The following table lists Amazon S3 error codes.

Error Code	Description	HTTP Status Code	SOAP Fault Code Prefix
`AccessDenied`	Access Denied	403 Forbidden	Client
`AccountProblem`	There is a problem with your AWS account that prevents the operation from completing successfully. Please use Contact Us.	403 Forbidden	Client
`AmbiguousGrantByEmailAddress`	The email address you provided is associated with more than one account.	400 Bad Request	Client
`BadDigest`	The Content-MD5 you specified did not match what we received.	400 Bad Request	Client

Error Code	Description	HTTP Status Code	SOAP Fault Code Prefix
BucketAlreadyExists	The requested bucket name is not available. The bucket namespace is shared by all users of the system. Please select a different name and try again.	409 Conflict	Client
BucketAlreadyOwnedByYou	Your previous request to create the named bucket succeeded and you already own it. You get this error in all AWS regions except US Standard, us-east-1. In us-east-1 region, you will get 200 OK, but it is no-op (if bucket exists it Amazon S3 will not do anything).	409 Conflict (in all regions except US Standard).	Client
BucketNotEmpty	The bucket you tried to delete is not empty.	409 Conflict	Client
CredentialsNotSupported	This request does not support credentials.	400 Bad Request	Client
CrossLocationLoggingProhibited	Cross-location logging not allowed. Buckets in one geographic location cannot log information to a bucket in another location.	403 Forbidden	Client
EntityTooSmall	Your proposed upload is smaller than the minimum allowed object size.	400 Bad Request	Client
EntityTooLarge	Your proposed upload exceeds the maximum allowed object size.	400 Bad Request	Client
ExpiredToken	The provided token has expired.	400 Bad Request	Client
IllegalVersioningConfigurationException	Indicates that the versioning configuration specified in the request is invalid.	400 Bad Request	Client
IncompleteBody	You did not provide the number of bytes specified by the Content-Length HTTP header	400 Bad Request	Client
IncorrectNumberOfFilesInPostRequest	POST requires exactly one file upload per request.	400 Bad Request	Client
InlineDataTooLarge	Inline data exceeds the maximum allowed size.	400 Bad Request	Client
InternalError	We encountered an internal error. Please try again.	500 Internal Server Error	Server

Error Code	Description	HTTP Status Code	SOAP Fault Code Prefix
`InvalidAccessKeyId`	The AWS access key Id you provided does not exist in our records.	403 Forbidden	Client
`InvalidAddressingHeader`	You must specify the Anonymous role.	N/A	Client
`InvalidArgument`	Invalid Argument	400 Bad Request	Client
`InvalidBucketName`	The specified bucket is not valid.	400 Bad Request	Client
`InvalidBucketState`	The request is not valid with the current state of the bucket.	409 Conflict	Client
`InvalidDigest`	The Content-MD5 you specified is not valid.	400 Bad Request	Client
`InvalidEncryptionAlgorithmError`	The encryption request you specified is not valid. The valid value is AES256.	400 Bad Request	Client
`InvalidLocationConstraint`	The specified location constraint is not valid. For more information about Regions, see How to Select a Region for Your Buckets.	400 Bad Request	Client
`InvalidObjectState`	The operation is not valid for the current state of the object.	403 Forbidden	Client
`InvalidPart`	One or more of the specified parts could not be found. The part might not have been uploaded, or the specified entity tag might not have matched the part's entity tag.	400 Bad Request	Client
`InvalidPartOrder`	The list of parts was not in ascending order.Parts list must specified in order by part number.	400 Bad Request	Client
`InvalidPayer`	All access to this object has been disabled.	403 Forbidden	Client
`InvalidPolicyDocument`	The content of the form does not meet the conditions specified in the policy document.	400 Bad Request	Client
`InvalidRange`	The requested range cannot be satisfied.	416 Requested Range Not Satisfiable	Client
`InvalidRequest`	SOAP requests must be made over an HTTPS connection.	400 Bad Request	Client
`InvalidSecurity`	The provided security credentials are not valid.	403 Forbidden	Client

Error Code	Description	HTTP Status Code	SOAP Fault Code Prefix
InvalidSOAPRequest	The SOAP request body is invalid.	400 Bad Request	Client
InvalidStorageClass	The storage class you specified is not valid.	400 Bad Request	Client
InvalidTargetBucketForLogging	The target bucket for logging does not exist, is not owned by you, or does not have the appropriate grants for the log-delivery group.	400 Bad Request	Client
InvalidToken	The provided token is malformed or otherwise invalid.	400 Bad Request	Client
InvalidURI	Couldn't parse the specified URI.	400 Bad Request	Client
KeyTooLong	Your key is too long.	400 Bad Request	Client
MalformedACLError	The XML you provided was not well-formed or did not validate against our published schema.	400 Bad Request	Client
MalformedPOSTRequest	The body of your POST request is not well-formed multipart/form-data.	400 Bad Request	Client
MalformedXML	This happens when the user sends malformed xml (xml that doesn't conform to the published xsd) for the configuration. The error message is, "The XML you provided was not well-formed or did not validate against our published schema."	400 Bad Request	Client
MaxMessageLengthExceeded	Your request was too big.	400 Bad Request	Client
MaxPostPreDataLengthExceededError	Your POST request fields preceding the upload file were too large.	400 Bad Request	Client
MetadataTooLarge	Your metadata headers exceed the maximum allowed metadata size.	400 Bad Request	Client
MethodNotAllowed	The specified method is not allowed against this resource.	405 Method Not Allowed	Client
MissingAttachment	A SOAP attachment was expected, but none were found.	N/A	Client
MissingContentLength	You must provide the Content-Length HTTP header.	411 Length Required	Client

Error Code	Description	HTTP Status Code	SOAP Fault Code Prefix
MissingRequestBodyError	This happens when the user sends an empty xml document as a request. The error message is, "Request body is empty."	400 Bad Request	Client
MissingSecurityElement	The SOAP 1.1 request is missing a security element.	400 Bad Request	Client
MissingSecurityHeader	Your request is missing a required header.	400 Bad Request	Client
NoLoggingStatusForKey	There is no such thing as a logging status subresource for a key.	400 Bad Request	Client
NoSuchBucket	The specified bucket does not exist.	404 Not Found	Client
NoSuchKey	The specified key does not exist.	404 Not Found	Client
NoSuchLifecycleConfiguration	The lifecycle configuration does not exist.	404 Not Found	Client
NoSuchUpload	The specified multipart upload does not exist. The upload ID might be invalid, or the multipart upload might have been aborted or completed.	404 Not Found	Client
NoSuchVersion	Indicates that the version ID specified in the request does not match an existing version.	404 Not Found	Client
NotImplemented	A header you provided implies functionality that is not implemented.	501 Not Implemented	Server
NotSignedUp	Your account is not signed up for the Amazon S3 service. You must sign up before you can use Amazon S3. You can sign up at the following URL: http://aws.amazon.com/s3	403 Forbidden	Client
NotSuchBucketPolicy	The specified bucket does not have a bucket policy.	404 Not Found	Client
OperationAborted	A conflicting conditional operation is currently in progress against this resource. Try again.	409 Conflict	Client
PermanentRedirect	The bucket you are attempting to access must be addressed using the specified endpoint. Send all future requests to this endpoint.	301 Moved Permanently	Client
PreconditionFailed	At least one of the preconditions you specified did not hold.	412 Precondition Failed	Client

Error Code	Description	HTTP Status Code	SOAP Fault Code Prefix
`Redirect`	Temporary redirect.	307 Moved Temporarily	Client
`RestoreAlreadyInProgress`	Object restore is already in progress.	409 Conflict	Client
`RequestIsNotMultiPartContent`	Bucket POST must be of the enclosure-type multipart/form-data.	400 Bad Request	Client
`RequestTimeout`	Your socket connection to the server was not read from or written to within the timeout period.	400 Bad Request	Client
`RequestTimeTooSkewed`	The difference between the request time and the server's time is too large.	403 Forbidden	Client
`RequestTorrentOfBucketError`	Requesting the torrent file of a bucket is not permitted.	400 Bad Request	Client
`SignatureDoesNotMatch`	The request signature we calculated does not match the signature you provided. Check your AWS secret access key and signing method. For more information, see REST Authentication and SOAP Authentication for details.	403 Forbidden	Client
`ServiceUnavailable`	Reduce your request rate.	503 Service Unavailable	Server
`SlowDown`	Reduce your request rate.	503 Slow Down	Server
`TemporaryRedirect`	You are being redirected to the bucket while DNS updates.	307 Moved Temporarily	Client
`TokenRefreshRequired`	The provided token must be refreshed.	400 Bad Request	Client
`TooManyBuckets`	You have attempted to create more buckets than allowed.	400 Bad Request	Client
`UnexpectedContent`	This request does not support content.	400 Bad Request	Client
`UnresolvableGrantByEmailAddress`	The email address you provided does not match any account on record.	400 Bad Request	Client
`UserKeyMustBeSpecified`	The bucket POST must contain the specified field name. If it is specified, check the order of the fields.	400 Bad Request	Client

REST Error Responses

When there is an error, the header information contains:

- Content-Type: application/xml
- An appropriate 3xx, 4xx, or 5xx HTTP status code

The body or the response also contains information about the error. The following sample error response shows the structure of response elements common to all REST error responses.

```
<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>NoSuchKey</Code>
  <Message>The resource you requested does not exist</Message>
  <Resource>/mybucket/myfoto.jpg</Resource>
  <RequestId>4442587FB7D0A2F9</RequestId>
</Error>
```

The following table explains the REST error response elements

Name	Description
Code	The error code is a string that uniquely identifies an error condition. It is meant to be read and understood by programs that detect and handle errors by type. For more information, see List of Error Codes (p. 3). Type: String Ancestor: Error
Error	Container for all error elements. Type: Container Ancestor: None
Message	The error message contains a generic description of the error condition in English. It is intended for a human audience. Simple programs display the message directly to the end user if they encounter an error condition they don't know how or don't care to handle. Sophisticated programs with more exhaustive error handling and proper internationalization are more likely to ignore the error message. Type: String Ancestor: Error
RequestId	ID of the request associated with the error. Type: String Ancestor: Error
Resource	The bucket or object that is involved in the error. Type: String Ancestor: Error

Many error responses contain additional structured data meant to be read and understood by a developer diagnosing programming errors. For example, if you send a Content-MD5 header with a REST PUT request that doesn't match the digest calculated on the server, you receive a BadDigest error. The error response also includes as detail elements the digest we calculated, and the digest you told us to expect. During

development, you can use this information to diagnose the error. In production, a well-behaved program might include this information in its error log.

For information about general response elements, go to Error Responses.

REST API

Topics

- Common Request Headers (p. 12)
- Common Response Headers (p. 14)
- Authenticating Requests (AWS Signature Version 4) (p. 15)
- Authenticating Requests in Browser-Based Uploads Using POST (AWS Signature Version 4) (p. 51)
- Operations on the Service (p. 65)
- Operations on Buckets (p. 68)
- Operations on Objects (p. 198)

This section contains information specific to the Amazon S3 REST API.

The examples in this guide use the newer virtual hosted-style method for accessing buckets instead of the path-style. Although the path-style is still supported for legacy applications, we recommend using the virtual-hosted style where applicable. For more information, see Working with Amazon S3 Buckets

The following example is a virtual hosted-style request that deletes the `puppy.jpg` file from the `mybucket` bucket.

```
DELETE /puppy.jpg HTTP/1.1
User-Agent: dotnet
Host: mybucket.s3.amazonaws.com
Date: Tue, 15 Jan 2008 21:20:27 +0000
x-amz-date: Tue, 15 Jan 2008 21:20:27 +0000
Authorization: signatureValue
```

The following example is a path-style version of the same request.

```
DELETE /mybucket/puppy.jpg HTTP/1.1
User-Agent: dotnet
Host: s3.amazonaws.com
Date: Tue, 15 Jan 2008 21:20:27 +0000
x-amz-date: Tue, 15 Jan 2008 21:20:27 +0000
Authorization: signatureValue
```

Common Request Headers

The following table describes headers that can be used by various types of Amazon S3 REST requests.

Header Name	Description
`Authorization`	The information required for request authentication. For more information, go to The Authentication Header in the *Amazon Simple Storage Service Developer Guide*. For anonymous requests this header is not required.
`Content-Length`	Length of the message (without the headers) according to RFC 2616. This header is required for PUTs and operations that load XML, such as logging and ACLs.
`Content-Type`	The content type of the resource in case the request content in the body. Example: `text/plain`
`Content-MD5`	The base64 encoded 128-bit MD5 digest of the message (without the headers) according to RFC 1864. This header can be used as a message integrity check to verify that the data is the same data that was originally sent. Although it is optional, we recommend using the Content-MD5 mechanism as an end-to-end integrity check. For more information about REST request authentication, go to REST Authentication in the *Amazon Simple Storage Service Developer Guide*.
`Date`	The current date and time according to the requester. Example: `Wed, 01 Mar 2006 12:00:00 GMT`. When you specify the `Authorization` header, you must specify either the `x-amz-date` or the `Date` header
`Expect`	When your application uses 100-continue, it does not send the request body until it receives an acknowledgment. If the message is rejected based on the headers, the body of the message is not sent. This header can be used only if you are sending a body. Valid Values: 100-continue
`Host`	For path-style requests, the value is `s3.amazonaws.com`. For virtual-style requests, the value is `BucketName.s3.amazonaws.com`. For more information, go to Virtual Hosting in the Amazon Simple Storage Service Developer Guide. This header is required for HTTP 1.1 (most toolkits add this header automatically); optional for HTTP/1.0 requests.
`x-amz-content-sha256`	When using signature version 4 to authenticate request, this header provides a hash of the request payload. For more information see Authenticating Requests by Using the Authorization Header (Compute Checksum of the Entire Payload Prior to Transmission) - Signature Version 4 (p. 19). When uploading object in chunks, you set the value to STREAMING-AWS4-HMAC-SHA256-PAYLOAD to indicate that the signature covers only headers and that there is no payload. For more information, see Authenticating Requests Using HTTP Authorization Header (Chunked Upload) (p. 31).

Header Name	Description
`x-amz-date`	The current date and time according to the requester. Example: `Wed, 01 Mar 2006 12:00:00 GMT`. When you specify the `Authorization` header, you must specify either the `x-amz-date` or the `Date` header. If you specify both, the value specified for the `x-amz-date` header takes precedence.
`x-amz-security-token`	This header can be used in the following scenarios: • **Provide security tokens for Amazon DevPay operations—**Each request that uses Amazon DevPay requires two `x-amz-security-token` headers: one for the product token and one for the user token. When Amazon S3 receives an authenticated request, it compares the computed signature with the provided signature. Improperly formatted multi-value headers used to calculate a signature can cause authentication issues • **Provide security token when using temporary security credentials—**When making requests using temporary security credentials you obtained from IAM you must provide a security token using this header. To learn more about temporary security credentials, go to Making Requests. This header is required for requests that use Amazon DevPay and requests that are signed using temporary security credentials.

Common Response Headers

The following table describes response headers that are common to most AWS S3 responses.

Name	Description
`Content-Length`	The length in bytes of the body in the response. Type: String Default: None
`Content-Type`	The MIME type of the content. For example, Content-Type: text/html; charset=utf-8 Type: String Default: None
`Connection`	specifies whether the connection to the server is open or closed. Type: Enum Valid Values: open | close Default: None
`Date`	The date and time Amazon S3 responded, for example, Wed, 01 Mar 2006 12:00:00 GMT. Type: String Default: None
`ETag`	The entity tag is a hash of the object. The ETag only reflects changes to the contents of an object, not its metadata. The ETag is determined when an object is created. For objects created by the PUT Object operation and the POST Object operation, the ETag is a quoted, 32-digit hexadecimal string representing the MD5 digest of the object data. For other objects, the ETag may or may not be an MD5 digest of the object data. If the ETag is not an MD5 digest of the object data, it will contain one or more non-hexadecimal characters and/or will consist of less than 32 or more than 32 hexadecimal digits. Type: String
`Server`	The name of the server that created the response. Type: String Default: AmazonS3
`x-amz-delete-marker`	Specifies whether the object returned was (true) or was not (false) a delete marker. Type: Boolean Valid Values: true | false Default: false
`x-amz-id-2`	A special token that helps AWS troubleshoot problems. Type: String Default: None
`x-amz-request-id`	A value created by Amazon S3 that uniquely identifies the request. In the unlikely event that you have problems with Amazon S3, AWS can use this value to troubleshoot the problem. Type: String Default: None

Name	Description
`x-amz-version-id`	The version of the object. When you enable versioning, Amazon S3 generates a random number for objects added to a bucket. The value is UTF-8 encoded and URL ready. When you PUT an object in a bucket where versioning has been suspended, the version ID is always `null`. Type: String Valid Values: null | any URL-ready, UTF-8 encoded string Default: null

Authenticating Requests (AWS Signature Version 4)

Topics

- Authentication Methods (p. 16)
- Introduction to Signing Requests (p. 16)
- Authenticating a Request in the Authorization Header (p. 17)
- Authenticating Requests by Using Query Parameters (AWS Signature Version 4) (p. 38)
- Examples: Signature Calculations in AWS Signature Version 4 (p. 43)
- Authenticating Requests in Browser-Based Uploads Using POST (AWS Signature Version 4) (p. 45)
- Amazon S3 Signature Version 4 Authentication Specific Policy Keys (p. 47)

Every interaction with Amazon S3 is either authenticated or anonymous. This section explains request authentication with the AWS Signature Version 4 algorithm.

Amazon S3 supports Signature Version 4, a protocol for authenticating inbound API requests to AWS services, in all AWS regions. At this time, existing AWS regions continue to support the previous protocol, Signature Version 2. Any new regions after January 30, 2014 will support only Signature Version 4. Note that Signature Version 4 requires the request body to be signed for added security. This requirement creates additional computation load that is not required in Signature Version 2. For more information about AWS Signature Version 2, go to Signing and Authenticating REST Requests in the *Amazon Simple Storage Service Developer Guide*.

> **Note**
> If you use the AWS SDKs (see Sample Code and Libraries) to send your requests, you don't need to read this section: the SDK clients authenticate your requests by using access keys that you provide. Unless you have a good reason not to, you should always use the AWS SDKs. In regions that support both signature versions, you can request AWS SDKs to use specific signature version. For more information, go to Specifying Signature Version in Request Authentication in the *Amazon Simple Storage Service Developer Guide*. You need to read this section only if you are implementing the AWS Signature Version 4 algorithm in your custom client.

Authentication enables the following:

- **Verification of the identity of the requester** – Authenticated requests require a signature that you create by using your access keys (access key ID, secret access key). For information about getting access keys, see How Do I Get Security Credentials? in the *AWS General Reference*. If you are using temporary security credentials, the signature calculations also require a security token. For more information, go to Creating Temporary Security Credentials in the *AWS Security Token Service* documentation.
- **In-transit data protection** – In order to prevent tampering with a request while it is in transit, you use some of the request elements to calculate the request signature. Upon receiving the request, Amazon

S3 calculates the signature by using the same request elements. If any request component received by Amazon S3 does not match the component that was used to calculate the signature, Amazon S3 will reject the request.

- **Protect against potential replay attacks** – A request must reach Amazon S3 within 15 minutes of the timestamp in the request; otherwise, Amazon S3 denies the request.

Authentication Methods

You can express authentication information by using one of the following methods:

- **HTTP Authorization header** – Using the HTTP Authorization header is the most common method of signing an Amazon S3 request. All the Amazon S3 REST operations except for browser-based uploads using POST requests include this header.

  When you create objects, you upload data in your request. When uploading data, you include a checksum of the payload in the signature calculation. You have two options:

  - You can compute a checksum of the entire payload prior to transmission; however, you might find this method inefficient for large uploads.
  - For large uploads, you can upload an object in chunks. Each individual chunk includes both the chunk data and some overhead, for example, the chunk size and the chunk signature. To calculate the chunk signature, you use both the chunk data and the signature of the previous chunk. By eliminating the need to read the payload twice—once for calculating a signature and once for performing the upload—chunked upload can significantly improve performance for large payloads. Chunked upload works in all cases, regardless of payload size, and so you can use chunked upload in all your uploads.

  For more information, see Authenticating a Request in the Authorization Header (p. 17).
- **Query string parameters** – You can use a query string to express a request entirely in a URL. In this case, you use query parameters to provide request information, including the authentication information. Because the request signature is part of the URL, this type of URL is often referred to as a presigned URL. You can use presigned URLs to embed clickable links, which can be valid for up to seven days, in HTML. For more information, see Authenticating Requests by Using Query Parameters (AWS Signature Version 4) (p. 38).

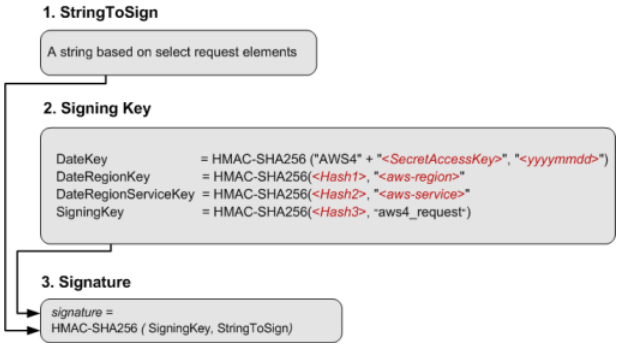
Amazon S3 also supports browser-based uploads that use an HTTP POST requests. With an HTTP POST request, you can upload content to Amazon S3 directly from the browser. For information about authenticating POST requests, go to Browser-Based Uploads Using POST in the *Amazon Simple Storage Service Developer Guide*.

Introduction to Signing Requests

Authentication information that you send in a request must include a signature. To calculate a signature, you first concatenate select request elements to form a string, referred to as the *string to sign*. You then use a signing key to calculate the hash-based message authentication code (HMAC) of the string to sign.

In AWS Signature Version 4, you don't use your secret access key to sign the request. Instead, you first use your secret access key to create a signing key. The signing key is scoped to a specific region and service. Additionally, the signing key expires seven days after creation. Because the scope and lifetime of the signing key are limited, your data is less at risk if the signing key is compromised.

The following diagram illustrates the general process of computing a signature.



The string to sign depends on the request type. For example, when you use the HTTP Authorization header or the query parameters for authentication, you use a varying combination of request elements to create the string to sign. For an HTTP POST request, the POST policy in the request is the string you sign. For more information about creating strings to sign, click one of the following topic links.

Upon receiving an authenticated request, Amazon S3 servers re-create the signature by using the authentication information that is contained in the request. If the signatures match, Amazon S3 will process your request; otherwise, the request will be rejected.

For more information about authenticating requests, see the following topics:

- Authenticating a Request in the Authorization Header (p. 17)
- Authenticating Requests by Using Query Parameters (AWS Signature Version 4) (p. 38)
- Authenticating Requests in Browser-Based Uploads Using POST (AWS Signature Version 4) (p. 51)

Authenticating a Request in the Authorization Header

Topics

- Overview (p. 17)
- Authenticating Requests by Using the Authorization Header (Compute Checksum of the Entire Payload Prior to Transmission) - Signature Version 4 (p. 19)
- Authenticating Requests Using HTTP Authorization Header (Chunked Upload) (p. 31)

Overview

Using the HTTP `Authorization` header is the most common method of providing authentication information. Except for POST requests (p. 238) and requests that are signed by using query parameters, all Amazon S3 bucket operations (p. 68) and object operations (p. 198) use the Authorization request header to provide authentication information.

The following is an example of the `Authorization` header value. Line breaks are added to this example for readability:

```
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-
east-1/s3/aws4_request,
SignedHeaders=host;range;x-amz-date,
Signature=fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc963176630326f1024
```

Note that there is space between the first two components, `AWS4-HMAC-SHA256` and `Credential`, and that the subsequent components, `Credential`, `SignedHeaders`, and `Signature` are separated by a comma.

The following table describes the various components of the Authorization header value.

Component	Description
`AWS4-HMAC-SHA256`	The algorithm that was used to calculate the signature. You must provide this value when you use AWS Signature Version 4 for authentication. The string specifies AWS Signature Version 4 (AWS4) and the signing algorithm (HMAC-SHA256).
`Credential`	Your access key ID and the scope information, which includes the date, region, and service that were used to calculate the signature. This string has the following form: *<your-access-key-id>*/*<date>*/*<aws-region>*/*<aws-service>*/aws4_request where • *<date>* value is specified using `yyyyMMdd` format. • *<aws-service>* value is `s3` when sending request to Amazon S3.
`SignedHeaders`	A semicolon-separated list of request headers that you used to compute `Signature`. The list includes header names only, and the header names must be in lowercase. For example, `host;range;x-amz-date`
`Signature`	The 256-bit signature expressed as 64 lowercase hexadecimal characters. For example, `fe5f80f77d5fa3be ca038a248ff027d0445342fe2855ddc963176630326f1024`

Upon receiving the request, Amazon S3 re-creates the string to sign using information in the `Authorization` header and the date header. It then verifies with authentication service the signatures match. The request date can be specified by using either the HTTP `Date` or the `x-amz-date` header. If both headers are present, `x-amz-date` takes precedence.

If the signatures match, Amazon S3 will process your request; otherwise, your request will fail.

If you use the Authorization header, you have the following options for transferring the payload :

- **Compute checksum of the entire payload prior to transmission** – The string that you construct for signing includes, among other things, a hash of the entire payload. For more information, see Authenticating Requests by Using the Authorization Header (Compute Checksum of the Entire Payload Prior to Transmission) - Signature Version 4 (p. 19).

If you sign the entire payload, you must read the file twice or buffer it in memory. For example, in order to upload a file, you will need to read the file first to compute a payload hash for signature calculation and again for transmission when you create the request. For smaller payloads, this approach might be preferable; however, for large files, reading the file twice can be inefficient, and so you might want to upload data in chunks.

- **Transfer payload in chunks** – You can break up your payload into chunks. These can be fixed or variable-size chunks. By uploading data in chunks, you avoid reading the entire payload to calculate the signature. Instead, for the first chunk, you calculate a seed signature that uses only the request headers. The second chunk contains the signature for the first chunk, and each subsequent chunk contains the signature for the chunk that precedes it. At the end of the upload, you send a final chunk with 0 bytes of data that contains the signature of the last chunk of the payload. For more information, see Authenticating Requests Using HTTP Authorization Header (Chunked Upload) (p. 31).

  You can transfer a payload in chunks regardless of the payload size.

For more information about these alternatives and related signature calculation, see the following topics:

- Authenticating Requests by Using the Authorization Header (Compute Checksum of the Entire Payload Prior to Transmission) - Signature Version 4 (p. 19)
- Authenticating Requests Using HTTP Authorization Header (Chunked Upload) (p. 31)

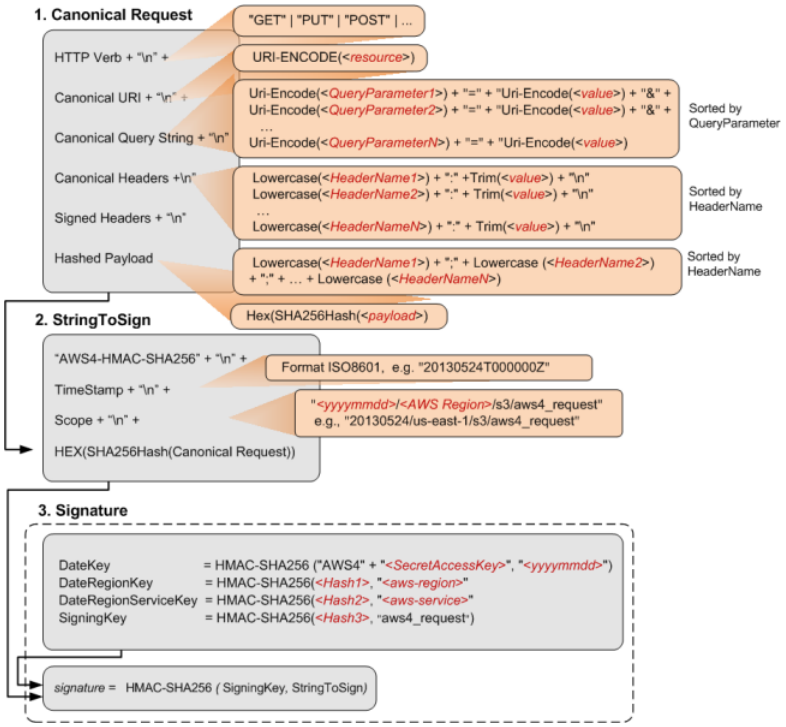
Authenticating Requests by Using the Authorization Header (Compute Checksum of the Entire Payload Prior to Transmission) - Signature Version 4

As described in the Overview (p. 17), for small payloads you might find it suitable to compute the payload hash prior to initiating the request. This section describes the signature calculation when you precompute a hash of your payload.

Calculating a Signature

To calculate a signature, you first need a string to sign. You then calculate a HMAC-SHA256 hash of the string to sign by using a signing key. The following diagram illustrates the process, including the various components of the string that you create for signing

When Amazon S3 receives an authenticated request, it computes the signature and then compares it with the signature that you provided in the request. For that reason, you must compute the signature by using the same method that is used by Amazon S3. The process of putting a request in an agreed-upon form for signing is called canonicalization.



The following table describes the functions that are shown in the diagram. You will need to implement code for these functions.

Description
Convert the string to lowercase.
Lowercase base 16 encoding.
Secure Hash Algorithm (SHA) cryptographic hash function.
Remove any leading or trailing whitespace.

	Description
	URI encode every byte. Uri-Encode() must enforce the following rules:

- URI encode every byte except the unreserved characters: 'A'-'Z', 'a'-'z', '0'-'9', '-', '.', '_', and '~'.
- The space character is a reserved character and must be encoded as "%20" (and not as "+").
- Each Uri-encoded byte is formed by a '%' and the two-digit hexadecimal value of the byte.
- Letters in the hexadecimal value must be uppercase, for example "%1A".
- Encode the forward slash character, '/', everywhere except in the object key name. For example, if the object key name is `photos/Jan/sample.jpg`, the forward slash in the key name is not encoded.

Caution
Standard Uri-Encode functions provided by your development platform may not work because of differences in implementation and related ambiguity in the underlying RFCs. We recommend that you write your own custom Uri-Encode function to ensure that your encoding will work.

The following is an example uri-encode() function in Java.

```
public static String uri-encode(CharSequence input, boolean encodeSlash) {
          StringBuilder result = new StringBuilder();
          for (int i = 0; i < input.length(); i++) {
              char ch = input.charAt(i);
              if ((ch >= 'A' && ch <= 'Z') || (ch >= 'a' && ch <= 'z') || (ch
 >= '0' && ch <= '9') || ch == '_' || ch == '-' || ch == '~' || ch == '.') {
                  result.append(ch);
              } else if (ch == '/') {
                  result.append(encodeSlash ? "%2F" : ch);
              } else {
                  result.append(toHexUTF8(ch));
              }
          }
          return result.toString();
      }
```

Task 1: Create a Canonical Request

This section provides an overview of creating a canonical request.

The following is the canonical request format that Amazon S3 uses to calculate a signature. For signatures to match, you must create a canonical request in this format:

```
<HTTPMethod>\n
<CanonicalURI>\n
<CanonicalQueryString>\n
<CanonicalHeaders>\n
<SignedHeaders>\n
<HashedPayload>
```

Where

- *HTTPMethod* is one of the HTTP methods, for example GET, PUT, HEAD, and DELETE.

- *CanonicalURI* is the URI-encoded version of the absolute path component of the URI—everything starting with the "/" that follows the domain name and up to the end of the string or to the question mark character ('?') if you have query string parameters. For example, in the URI

```
http://s3.amazonaws.com/examplebucket/myphoto.jpg
```

  /examplebucket/myphoto.jpg is the absolute path. In the absolute path, you don't encode the "/".
- *CanonicalQueryString* specifies the URI-encoded query string parameters. You URI-encode name and values individually. You must also sort the parameters in the canonical query string alphabetically by key name. The sorting occurs after encoding. For example, in the URI

```
http://s3.amazonaws.com/examplebucket?prefix=somePrefix&marker=someMarker&max-keys=20
```

  the query string is prefix=somePrefix&marker=someMarker&max-keys=20. The canonical query string is as follows. Line breaks are added to this example for readability:

```
URI-encode("marker")+"="+URI-encode("someMarker")+"&"+
URI-encode("max-keys")+"="+URI-encode("20") + "&" +
URI-encode("prefix")+"="+URI-encode("somePrefix")
```

  When a request targets a subresource, the corresponding query parameter value will be an empty string (""). For example, the following URI identifies the ACL subresource on the examplebucket bucket.

```
http://s3.amazonaws.com/examplebucket?acl
```

  The CanonicalQueryString in this case is:

```
URI-encode("acl") + "=" + ""
```

  If the URI does not include a '?', there is no query string in the request, and you set the canonical query string to an empty string (""). You will still need to include the "\n".
- *CanonicalHeaders* is a list of request headers with their values. Individual header name and value pairs are separated by the newline character ("\n"). Header names must be in lowercase. You must sort the header names alphabetically to construct the string, as shown in the following example:

```
Lowercase(<HeaderName1>)+":"+Trim(<value>)+"\n"
Lowercase(<HeaderName2>)+":"+Trim(<value>)+"\n"
...
Lowercase(<HeaderNameN>)+":"+Trim(<value>)+"\n"
```

  The Lowercase() and Trim() functions used in this example are described in the preceding section.

  The *CanonicalHeaders* list must include the following:
  - HTTP host header
  - If the Content-Type header is present in the request, it must be added to the *CanonicalHeaders* list.
  - Any x-amz-* headers that you plan to include in your request must also be added. For example, if you are using temporary security credentials, you will include x-amz-security-token in your request. You must add this header in the list of *CanonicalHeaders*.

The following is an example `CanonicalHeaders` string. The header names are in lowercase and sorted.

```
host:s3.amazonaws.com
x-amz-content-sha256:e3b0c44298fc1c149af
bf4c8996fb92427ae41e4649b934ca495991b785
2b855
x-amz-date:20130708T220855Z
```

Note
For the purpose of calculating a signature, only the host and any `x-amz-*` headers are required; however, in order to prevent data tampering, you should consider including all the headers in the signature calculation. The `x-amz-content-sha256` header in the previous example provides a hash of the request payload. If there is no payload, you provide the hash of an empty string.

- *`SignedHeaders`* is an alphabetically sorted, semicolon-separated list of lowercase request header names. The request headers in the list are the same headers that you included in the `CanonicalHeaders` string. For example, for the previous example, the value of *`SignedHeaders`* would be as follows:

```
host;x-amz-content-sha256;x-amz-date
```

- *`HashedPayload`* is the hexadecimal value of the SHA256 hash of the request payload.

```
Hex(SHA256Hash(<payload>)
```

If there is no payload in the request, you compute a hash of the empty string as follows:

```
Hex(SHA256Hash(""))
```

The hash returns the following value:

```
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

For example, when you upload an object by using a PUT request, you provide object data in the body. When you retrieve an object by using a GET request, you compute the empty string hash.

Task 2: Create a String to Sign

This section provides an overview of creating a string to sign. For step-by-step instructions, go to Task 2: Create a String to Sign in the *AWS General Reference*.

The string to sign is a concatenation of the following strings:

```
"AWS4-HMAC-SHA256" + \n" +
timeStampISO8601Format + "\n" +
<Scope> + "\n" +
Hex(SHA256Hash(<CanonicalRequest>))
```

The constant string `AWS4-HMAC-SHA256` specifies the hash algorithm that you are using, HMAC-SHA256. The `timeStamp` is the current UTC time in ISO 8601 format (for example, `20130524T000000Z`).

Scope binds the resulting signature to a specific date, an AWS region, and a service. Thus, your resulting signature will work only in the specific region and for a specific service. The signature is valid for seven days after the specified date.

```
date.Format(<yyyyMMdd>) + "/" + <region> + "/" + <service> + "/aws4_request"
```

For Amazon S3, the service string is s3. For a list of *region* strings, go to Regions and Endpoints in the *AWS General Reference*. The Region column in this table provides the list of valid region strings.

The following scope restricts the resulting signature to the us-east-1 region and Amazon S3.

```
20130606/us-east-1/s3/aws4_request
```

Note
Scope must use the same date that you use to compute the signing key, as discussed in the following section.

Task 3: Calculate Signature

In AWS Signature Version 4, instead of using your AWS access keys to sign a request, you first create a signing key that is scoped to a specific region and service. For more information about signing keys, see Introduction to Signing Requests (p. 16).

```
DateKey              = HMAC-SHA256("AWS4"+"<SecretAccessKey>", "<yyyymmdd>")
DateRegionKey        = HMAC-SHA256(<DateKey>, "<aws-region>")
DateRegionServiceKey = HMAC-SHA256(<DateRegionKey>, "<aws-service>")
SigningKey           = HMAC-SHA256(<DateRegionServiceKey>, "aws4_request")
```

Note
This signing key is valid for seven days from the date specified in the DateKey hash.

For a list of region strings, go to Regions and Endpoints in the *AWS General Reference*.

Using a signing key enables you to keep your AWS credentials in one safe place. For example, if you have multiple servers that communicate with Amazon S3, you share the signing key with those servers; you don't have to keep a copy of your secret access key on each server. Signing key is valid for up to seven days. So each time you calculate signing key you will need to share the signing key with your servers. For more information, see Authenticating Requests (AWS Signature Version 4) (p. 15).

The final signature is the HMAC-SHA256 hash of the string to sign, using the signing key as the key.

```
HMAC-SHA256(SigningKey, StringToSign)
```

For step-by-step instructions on creating a signature, go to Task 3: Create a Signature in the *AWS General Reference*.

Examples: Signature Calculations

You can use the examples in this section as a reference to check signature calculations in your code. For additional references, go to Signature Version 4 Test Suite of the *AWS General Reference*. The calculations shown in the examples use the following data:

- Example access keys.

Parameter	Value
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

- Request timestamp of 20130524T000000Z (Fri, 24 May 2013 00:00:00 GMT).
- Bucket name examplebucket.
- The bucket is assumed to be in the US Standard region. The credential Scope and the Signing Key calculations use us-east-1 as the region specifier. For information about other regions, go to Regions and Endpoints in the *AWS General Reference*.
- You can use either path-style or virtual hosted–style requests. The following examples show how to sign a virtual hosted–style request, for example:

```
https://examplebucket.s3.amazonaws.com/photos/photo1.jpg
```

For more information, go to Virtual Hosting of Buckets in the *Amazon Simple Storage Service Developer Guide*.

Example: GET Object

The following example gets the first 10 bytes of an object (test.txt) from examplebucket. For more information about the API action, see GET Object (p. 212).

```
GET /test.txt HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Date: Fri, 24 May 2013 00:00:00 GMT
Authorization: SignatureToBeCalculated
Range: bytes=0-9
x-amz-content-sha256:e3b0c44298fc1c149af
bf4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date: 20130524T000000Z
```

Because this GET request does not provide any body content, the x-amz-content-sha256 value is the hash of the empty request body. The following steps show signature calculations and construction of the Authorization header.

1. **StringToSign**

   a. **CanonicalRequest**

```
GET
/test.txt

host:examplebucket.s3.amazonaws.com
range:bytes=0-9
x-amz-content-sha256:e3b0c44298fc1c149af
bf4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20130524T000000Z

host;range;x-amz-content-sha256;x-amz-date
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

In the canonical request string, the last line is the hash of the empty request body. The third line is empty because there are no query parameters in the request.

b. **StringToSign**

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
7344ae5b7ee6c3e7e6b0fe0640412a37625d1fbfff95c48bbb2dc43964946972
```

2. **SigningKey**

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>","20130524"),"us-east-1"),"s3"),"aws4_request")
```

3. **Signature**

```
f0e8bdb87c964420e857bd35b5d6ed310bd44f0170aba48dd91039c6036bdb41
```

4. **Authorization header**

The resulting `Authorization` header is as follows:

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-
1/s3/aws4_request,SignedHeaders=host;range;x-amz-content-sha256;x-amz-
date,Signa
ture=f0e8bdb87c964420e857bd35b5d6ed310bd44f0170aba48dd91039c6036bdb41
```

Example: PUT Object

This example PUT request creates an object (`test$file.txt`) in `examplebucket` . The example assumes the following:

- You are requesting `REDUCED_REDUNDANCY` as the storage class by adding the `x-amz-storage-class` request header. For information about storage classes, go to Storage Classes in the Amazon Simple Storage Service Developer Guide.
- The content of the uploaded file is a string, "Welcome to Amazon S3." The value of `x-amz-content-sha256` in the request is based on this string.

For information about the API action, see PUT Object (p. 250).

```
PUT test$file.txt HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Date: Fri, 24 May 2013 00:00:00 GMT

Authorization: SignatureToBeCalculated
x-amz-date: 20130524T000000Z
x-amz-storage-class: REDUCED_REDUNDANCY
x-amz-content-sha256: 44ce7dd67c959e0d3524ffac1771df
bba87d2b6b4b4e99e42034a8b803f8b072
```

```
<Payload>
```

The following steps show signature calculations.

1. **StringToSign**

   a. **CanonicalRequest**

   ```
   PUT
   /test%24file.text

   date:Fri, 24 May 2013 00:00:00 GMT
   host:examplebucket.s3.amazonaws.com
   x-amz-content-sha256:44ce7dd67c959e0d3524ffac1771df
   bba87d2b6b4b4e99e42034a8b803f8b072
   x-amz-date:20130524T000000Z
   x-amz-storage-class:REDUCED_REDUNDANCY

   date;host;x-amz-content-sha256;x-amz-date;x-amz-storage-class
   44ce7dd67c959e0d3524ffac1771dfbba87d2b6b4b4e99e42034a8b803f8b072
   ```

   In the canonical request, the third line is empty because there are no query parameters in the request. The last line is the hash of the body, which should be same as the `x-amz-content-sha256 header` value.

   b. **StringToSign**

   ```
   AWS4-HMAC-SHA256
   20130524T000000Z
   20130524/us-east-1/s3/aws4_request
   9e0e90d9c76de8fa5b200d8c849cd5b8dc7a3be3951ddb7f6a76b4158342019d
   ```

2. **SigningKey**

   ```
   signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
   "<YourSecretAccessKey>","20130524"),"us-east-1"),"s3"),"aws4_request")
   ```

3. **Signature**

   ```
   98ad721746da40c64f1a55b78f14c238d841ea1380cd77a1b5971af0ece108bd
   ```

4. **Authorization header**

   The resulting `Authorization header` is as follows:

   ```
   AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-
   1/s3/aws4_request,SignedHeaders=date;host;x-amz-content-sha256;x-amz-date;x-
   amz-storage-class,Signa
   ture=98ad721746da40c64f1a55b78f14c238d841ea1380cd77a1b5971af0ece108bd
   ```

Example: GET Bucket Lifecycle

The following GET request retrieves the lifecycle configuration of `examplebucket`. For information about the API action, see GET Bucket lifecycle (p. 95).

```
GET ?lifecycle HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Authorization: SignatureToBeCalculated
x-amz-date: 20130524T000000Z
x-amz-content-sha256:e3b0c44298fc1c149af
bf4c8996fb92427ae41e4649b934ca495991b7852b855
```

Because the request does not provide any body content, the `x-amz-content-sha256` header value is the hash of the empty request body. The following steps show signature calculations.

1. **StringToSign**

   a. **CanonicalRequest**

   ```
   GET
   /
   lifecycle=
   host:examplebucket.s3.amazonaws.com
   x-amz-content-sha256:e3b0c44298fc1c149af
   bf4c8996fb92427ae41e4649b934ca495991b7852b855
   x-amz-date:20130524T000000Z

   host;x-amz-content-sha256;x-amz-date
   e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
   ```

   In the canonical request, the last line is the hash of the empty request body.

   b. **StringToSign**

   ```
   AWS4-HMAC-SHA256
   20130524T000000Z
   20130524/us-east-1/s3/aws4_request
   9766c798316ff2757b517bc739a67f6213b4ab36dd5da2f94eaebf79c77395ca
   ```

2. **SigningKey**

   ```
   signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
   "<YourSecretAccessKey>","20130524"),"us-east-1"),"s3"),"aws4_request")
   ```

3. **Signature**

   ```
   fea454ca298b7da1c68078a5d1bdbfbbe0d65c699e0f91ac7a200a0136783543
   ```

4. **Authorization header**

   The resulting `Authorization` header is as follows:

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-
1/s3/aws4_request,SignedHeaders=host;x-amz-content-sha256;x-amz-date,Signa
ture=fea454ca298b7da1c68078a5d1bdbfbbe0d65c699e0f91ac7a200a0136783543
```

Example: Get Bucket (List Objects)

The following example retrieves a list of objects from `examplebucket` bucket. For information about the API action, see GET Bucket (List Objects) (p. 81).

```
GET ?max-keys=2&prefix=J HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Authorization: SignatureToBeCalculated
x-amz-date: 20130524T000000Z
x-amz-content-sha256:e3b0c44298fc1c149af
bf4c8996fb92427ae41e4649b934ca495991b7852b855
```

Because the request does not provide a body, the value of `x-amz-content-sha256` is the hash of the empty request body. The following steps show signature calculations.

1. **StringToSign**

   a. **CanonicalRequest**

   ```
   GET
   /
   max-keys=2&prefix=J
   host:examplebucket.s3.amazonaws.com
   x-amz-content-sha256:e3b0c44298fc1c149af
   bf4c8996fb92427ae41e4649b934ca495991b7852b855
   x-amz-date:20130524T000000Z

   host;x-amz-content-sha256;x-amz-date
   e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
   ```

   In the canonical string, the last line is the hash of the empty request body.

   b. **StringToSign**

   ```
   AWS4-HMAC-SHA256
   20130524T000000Z
   20130524/us-east-1/s3/aws4_request
   df57d21db20da04d7fa30298dd4488ba3a2b47ca3a489c74750e0f1e7df1b9b7
   ```

2. **SigningKey**

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>","20130524"),"us-east-1"),"s3"),"aws4_request")
```

3. **Signature**

```
34b48302e7b5fa45bde8084f4b7868a86f0a534bc59db6670ed5711ef69dc6f7
```

4. **Authorization header**

   The resulting `Authorization` header is as follows:

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-
1/s3/aws4_request,SignedHeaders=host;x-amz-content-sha256;x-amz-date,Signa
ture=34b48302e7b5fa45bde8084f4b7868a86f0a534bc59db6670ed5711ef69dc6f7
```

Authenticating Requests Using HTTP Authorization Header (Chunked Upload)

As described in the Overview (p. 17), you have an option of uploading the payload in chunks. You can send data in fixed size or variable size chunks. This section describes the signature calculation process in chunked upload, how you create the chunk body, and how the delayed signing works where you first upload the chunk, and send its signature in the subsequent chunk.

> **Note**
> When transferring data in a series of chunks, you can use either the `Transfer-Encoding` or the `Content-Length` HTTP header. In the event the client you are using does not support the `Transfer-Encoding` header, you can add the `Content-Length` HTTP header to explicitly specify the total content length. This will require you to first compute the total length of the payload including the metadata you will send in each chunk. If you use the `Transfer-Encoding` you can transmit content before you know the total size of the payload.

Each chunk is a finite structure that includes chunk data and some metadata, such as chunk size and chunk signature. Before sending the first chunk, you create a signature, referred to as "seed signature", in which you compute the signature using only the headers. You assume the seed signature as the signature of the zeroth chunk (something that does not exist). You send this signature in the first chunk. For each subsequent chunk, you create a chunk signature in which the string to sign includes the signature of the previous chunk. Thus, the chunk signatures are chained together; that is, signature of chunk *n* is a function *F(chunk n, signature(chunk n-1))*. The chaining ensures you send the chunks in correct order.

To perform a chunked upload:

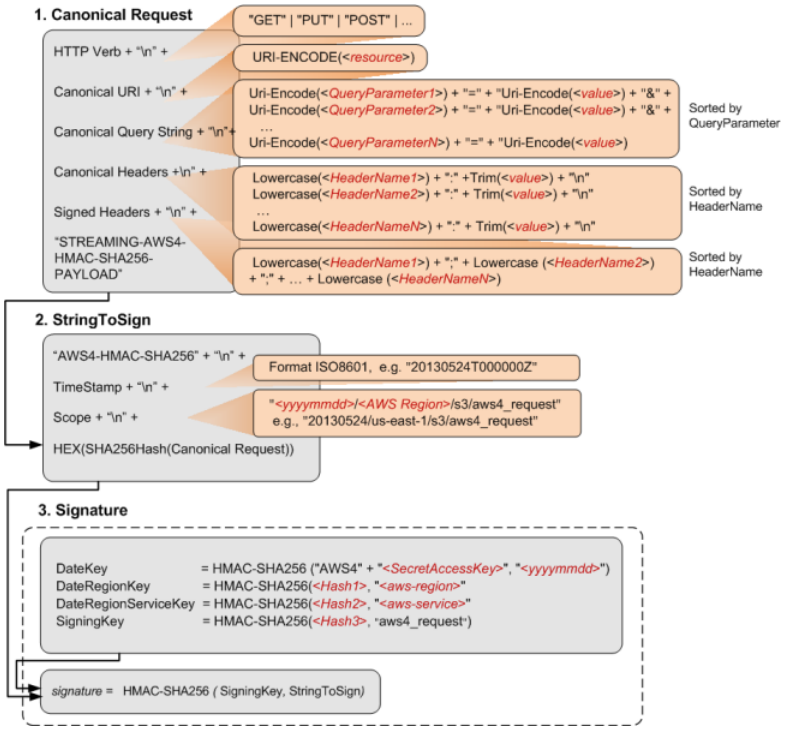
1. Decide payload chunk size. You will need this when you write the code.

   Chunk size must be at least 8 KB. We recommend a chunk size of a least 64 KB for better performance.

   This chunk size applies to all chunk except the last one. The last chunk you send can be smaller than 8 KB. If your payload is small and can fit in one chunk, then it can be smaller than the 8 KB.
2. Create the seed signature for inclusion in the first chunk. For more information, see Calculating the Seed Signature (p. 31).
3. Create first chunk and stream it. For more information, see Defining the Chunk Body (p. 34).
4. For each subsequent chunk, calculate the chunk signature that includes the previous signature in the string you sign, construct the chunk and send it. For more information, see Defining the Chunk Body (p. 34).
5. Send the final additional chunk, same as other chunks in construction but has zero data bytes. For more information, see Defining the Chunk Body (p. 34).

Calculating the Seed Signature

The following diagram illustrates the process of calculating the seed signature.



The following table describes the functions that are shown in the diagram. You will need to implement code for these functions.

Description
Convert the string to lowercase.
Lowercase base 16 encoding.
Secure Hash Algorithm (SHA) cryptographic hash function.
Remove any leading or trailing whitespace.

Description
URI encode every byte. Uri-Encode() must enforce the following rules: • URI encode every byte except the unreserved characters: 'A'-'Z', 'a'-'z', '0'-'9', '-', '.', '_', and '~'. • The space character is a reserved character and must be encoded as "%20" (and not as "+"). • Each Uri-encoded byte is formed by a '%' and the two-digit hexadecimal value of the byte. • Letters in the hexadecimal value must be uppercase, for example "%1A". • Encode the forward slash character, '/', everywhere except in the object key name. For example, if the object key name is `photos/Jan/sample.jpg`, the forward slash in the key name is not encoded. **Caution** Standard Uri-Encode functions provided by your development platform may not work because of differences in implementation and related ambiguity in the underlying RFCs. We recommend that you write your own custom Uri-Encode function to ensure that your encoding will work. The following is an example uri-encode() function in Java. `public static String uri-encode(CharSequence input, boolean encodeSlash) { StringBuilder result = new StringBuilder(); for (int i = 0; i < input.length(); i++) { char ch = input.charAt(i); if ((ch >= 'A' && ch <= 'Z') || (ch >= 'a' && ch <= 'z') || (ch >= '0' && ch <= '9') || ch == '_' || ch == '-' || ch == '~' || ch == '.') { result.append(ch); } else if (ch == '/') { result.append(encodeSlash ? "%2F" : ch); } else { result.append(toHexUTF8(ch)); } } return result.toString(); }`

For information about the signing process, see Authenticating Requests by Using the Authorization Header (Compute Checksum of the Entire Payload Prior to Transmission) - Signature Version 4 (p. 19). The process is the same except that the creation of `CanonicalRequest` differs as follows:

- In addition to the request headers you plan to add, you must include the following headers:

Header	Description
`x-amz-content-sha256`	Set the value to `STREAMING-AWS4-HMAC-SHA256-PAYLOAD` to indicate that the signature covers only headers and that there is no payload.
`Content-Encoding`	Set the value to `aws-chunked`. Amazon S3 supports multiple content encodings, for example, `Content-Encoding : aws-chunked, gzip` That is, you can specify your custom content-encoding when using Signature Version 4 streaming API.

Header	Description
x-amz-decoded-content-length	Set the value to the length, in bytes, of the data to be chunked, without counting any metadata. For example, if you are uploading a 4 GB file, set the value to 4294967296.
`Content-Length`	Set the value to the length of your data, including the metadata. Each chunk will have metadata, such as the signature of the previous chunk. Chunk calculations are discussed in the following section. **Note** If you don't want to calculate the length of the data including the metadata, you can instead use the `Transfer-Encoding` HTTP header with the value `chunked`. Amazon S3 supports this low-level alternative in the event the client you are using does not supports `Transfer-Encoding`.

You send the first chunk with the seed signature. You will need to construct the chunk as described in the following section.

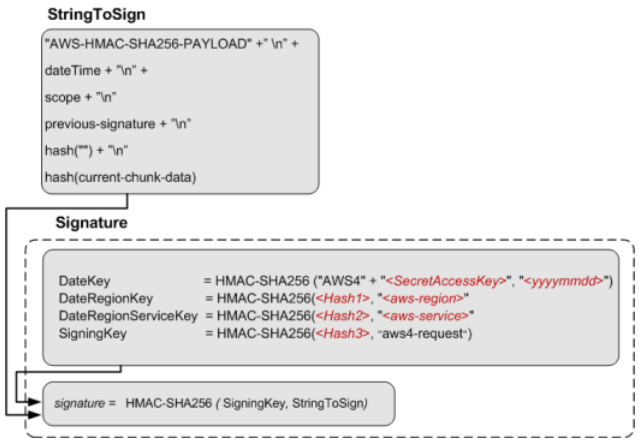
Defining the Chunk Body

All chunks include some metadata. Each chunk must conform to the following structure:

```
string(IntHexBase(chunk-size)) + ";chunk-signature=" + signature + \r\n + chunk-data + \r\n
```

Where

- `IntHexBase()` is a function that you will write to convert an integer chunk-size to hexadecimal. For example, if chunk-size is 65536, hexadecimal string is "1000".
- *`chunk-size`* is the size, in bytes, of the chunk-data, without metadata. For example, if you are uploading a 65 KB object and using a chunk size of 64 KB, you upload the data in three chunks: the first would be 64 KB, the second 1 KB, and the final chunk with 0 bytes.
- *`signature`* for the first chunk, you include the seed signature. For subsequent chunks, you calculate chunk signatures using the following string to sign. The string to sign includes the signature of the previous chunk, which is shown here as previous-signature.



The size of the final chunk data that you send is 0, although the chunk body will still store metadata, including the signature of the previous chunk.

Example: PUT Object

You can use the examples in this section as a reference to check signature calculations in your code. Before you review the examples, note the following:

- The signature calculations in these examples use the following example security credentials.

Parameter	Value
AWSAccessKeyId	`AKIAIOSFODNN7EXAMPLE`
AWSSecretAccessKey	`wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`

- All examples use the request timestamp 20130524T000000Z (`Fri, 24 May 2013 00:00:00 GMT`).
- All examples use `examplebucket` as the bucket name.
- The bucket is assumed to be in the US Standard region, and the credential `Scope` and the `Signing Key` calculations use `us-east-1` as the region specifier. For more information, go to Regions and Endpoints in the *Amazon Web Services General Reference*.
- You can use either path style or virtual-hosted style requests. The examples below show use virtual-hosted style requests, for example:

```
https://examplebucket.s3.amazonaws.com/photos/photo1.jpg
```

  For more information, go to Virtual Hosting of Buckets in the *Amazon Simple Storage Service Developer Guide*.

Example: PUT Object

The following example sends a PUT request to upload an object. The signature calculations assume the following:

- You are uploading a 65 KB text file, and the file content is a one-character string made up of the letter 'a'.
- The chunk size is 64 KB. As a result, the payload will be uploaded in three chunks, 64 KB, 1 KB, and the final chunk with 0 bytes of chunk data.
- The resulting object has the key name `chunkObject.txt`.
- You are requesting REDUCED_REDUNDANCY as the storage class by adding the `x-amz-storage-class` request header.

For information about the API action, see PUT Object (p. 250). The general request syntax is:

```
PUT /examplebucket/chunkObject.txt HTTP/1.1
Host: s3.amazonaws.com
x-amz-date: 20130524T000000Z
x-amz-storage-class: REDUCED_REDUNDANCY
Authorization: SignatureToBeCalculated
x-amz-content-sha256: STREAMING-AWS4-HMAC-SHA256-PAYLOAD
Content-Encoding: aws-chunked
x-amz-decoded-content-length: 66560
```

```
Content-Length: 66824
<Payload>
```

The following steps show signature calculations.

1. **Seed signature — Create String to Sign**

   1. **CanonicalRequest**

      ```
      PUT
      /examplebucket/chunkObject.txt

      content-encoding:aws-chunked
      content-length:66824
      host:s3.amazonaws.com
      x-amz-content-sha256:STREAMING-AWS4-HMAC-SHA256-PAYLOAD
      x-amz-date:20130524T000000Z
      x-amz-decoded-content-length:66560
      x-amz-storage-class:REDUCED_REDUNDANCY

      content-encoding;content-length;host;x-amz-content-sha256;x-amz-date;x-
      amz-decoded-content-length;x-amz-storage-class
      STREAMING-AWS4-HMAC-SHA256-PAYLOAD
      ```

      In the canonical request, the third line is empty because there are no query parameters in the request. The last line is the constant string provided as the value of the hashed Payload which should be same as the value of `x-amz-content-sha256 header`.

   2. **StringToSign**

      ```
      AWS4-HMAC-SHA256
      20130524T000000Z
      20130524/us-east-1/s3/aws4_request
      cee3fed04b70f867d036f722359b0b1f2f0e5dc0efadbc082b76c4c60e316455
      ```

2. **SigningKey**

   ```
   signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
   "<YourSecretAccessKey>","20130524"),"us-east-1"),"s3"),"aws4_request")
   ```

3. **Seed Signature**

   ```
   4f232c4386841ef735655705268965c44a0e4690baa4adea153f7db9fa80a0a9
   ```

4. **Authorization header**

   The resulting Authorization header is as follows:

   ```
   AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-
   1/s3/aws4_request,
   SignedHeaders=content-encoding;content-length;host;x-amz-content-sha256;x-
   ```

```
amz-date;x-amz-decoded-content-length;x-amz-storage-class,
Signature=4f232c4386841ef735655705268965c44a0e4690baa4adea153f7db9fa80a0a9
```

5. **Chunk 1: (65536 bytes, with value 97 for letter 'a')**

   1. Chunk string to sign:

      ```
      AWS4-HMAC-SHA256-PAYLOAD
      20130524T000000Z
      20130524/us-east-1/s3/aws4_request
      4f232c4386841ef735655705268965c44a0e4690baa4adea153f7db9fa80a0a9
      e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
      bf718b6f653bebc184e1479f1935b8da974d701b893afcf49e701f3e2f9f9c5a
      ```

   2. Chunk signature:

      ```
      ad80c730a21e5b8d04586a2213dd63b9a0e99e0e2307b0ade35a65485a288648
      ```

   3. Chunk data sent:

      ```
      10000;chunk-signa
      ture=ad80c730a21e5b8d04586a2213dd63b9a0e99e0e2307b0ade35a65485a288648
      <65536-bytes>
      ```

6. **Chunk 2: (1024 bytes, with value 97 for letter 'a')**

   1. Chunk string to sign:

      ```
      AWS4-HMAC-SHA256-PAYLOAD
      20130524T000000Z
      20130524/us-east-1/s3/aws4_request
      ad80c730a21e5b8d04586a2213dd63b9a0e99e0e2307b0ade35a65485a288648
      e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
      2edc986847e209b4016e141a6dc8716d3207350f416969382d431539bf292e4a
      ```

   2. Chunk signature:

      ```
      0055627c9e194cb4542bae2aa5492e3c1575bbb81b612b7d234b86a503ef5497
      ```

   3. Chunk data sent:

      ```
      400;chunk-signa
      ture=0055627c9e194cb4542bae2aa5492e3c1575bbb81b612b7d234b86a503ef5497
      <1024 bytes>
      ```

7. **Chunk 3: (0 byte data)**

   1. Chunk string to sign:

      ```
      AWS4-HMAC-SHA256-PAYLOAD
      20130524T000000Z
      20130524/us-east-1/s3/aws4_request
      0055627c9e194cb4542bae2aa5492e3c1575bbb81b612b7d234b86a503ef5497
      e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
      e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
      ```

   2. Chunk signature:

      ```
      b6c6ea8a5354eaf15b3cb7646744f4275b71ea724fed81ceb9323e279d449df9
      ```

   3. Chunk data sent:

      ```
      0;chunk-signa
      ture=b6c6ea8a5354eaf15b3cb7646744f4275b71ea724fed81ceb9323e279d449df9
      ```

Authenticating Requests by Using Query Parameters (AWS Signature Version 4)

Using query parameters to authenticate requests is useful when you want to express a request entirely in a URL. This method is also referred as presigning a URL. Presigned URLs enable you to grant temporary access to your Amazon S3 resources. The end user can then enter the presigned URL in his or her browser to access the specific Amazon S3 resource. You can also use presigned URLs to embed clickable links in HTML. For example, you might store videos in an Amazon S3 bucket and make them available on your website by using presigned URLs.

The following is an example presigned URL. The line feeds in the URL are added for readability.

```
https://s3.amazonaws.com/examplebucket/test.txt
?X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=<your-access-key-id>/20130721/us-east-1/s3/aws4_request
&X-Amz-Date=20130721T201207Z
&X-Amz-Expires=86400
&X-Amz-SignedHeaders=host
&X-Amz-Signature=<signature-value>
```

In the preceding example, the `X-Amz-Credential` value in the URL shows the "/" character only for readability; in practice, it should be encoded as `%2F`.

```
&X-Amz-Credential=<your-access-key-id>%2F20130721%2Fus-east-1%2Fs3%2Faws4_request
```

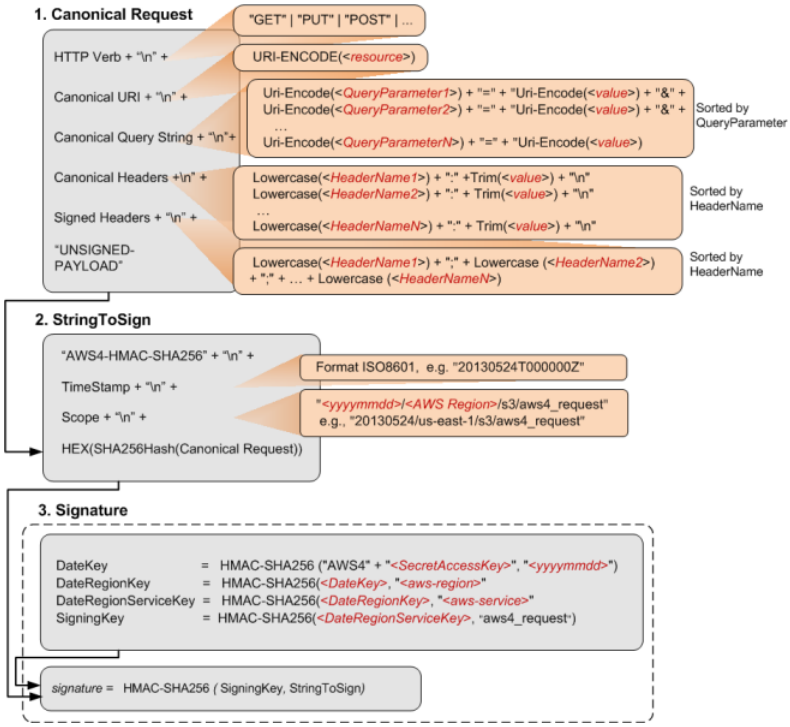
The URL includes a set of query parameters that provide authentication information, such as a signature and other information that helps Amazon S3 calculate the signature. If the signatures match, Amazon S3

will process your request. The following table describes the query parameters required in a presigned URL.

Query String Parameter Name	Example Value
X-Amz-Algorithm	Identifies the version of AWS Signature and the algorithm that you used to calculate the signature. For AWS Signature Version 4, you set this parameter value to "AWS4-HMAC-SHA256". This string identifies AWS Signature Version 4 (AWS4) and the HMAC-SHA256 algorithm (HMAC-SHA256).
X-Amz-Credential	In addition to your access key ID, this parameter also provides scope information identifying the region and service for which the signature is valid. This value should match the scope that you use to calculate the signing key, as discussed in the following section. The general form for this parameter value is as follows: *<your-access-key-id>*/*<date>*/*<AWS-region>*/*<AWS-service>*/aws4_request For example: AKIAIOSFODNN7EXAMPLE/20130721/us-east-1/s3/aws4_request. For Amazon S3, the *AWS-service* string is "s3". For a list of *AWS-region* strings, go to Regions and Endpoints in the *Amazon Web Services General Reference*
X-Amz-Date	The date in ISO 8601 format, for example, 20130721T201207Z. This value must match the date value used you use to calculate the signature.
X-Amz-Expires	Provides the time period, in seconds, for which the generated presigned URL is valid. For example, 86400 (24 hours). This value is an integer. The minimum value you can set is 1, and the maximum is 604800 (seven days). A presigned URL can be valid for a maximum of seven days because the signing key you use in signature calculation is valid for up to seven days.
X-Amz-SignedHeaders	Lists the headers that you used to calculate the signature. The HTTP host header is required. Any x-amz-* headers that you plan to add to the request are also required for signature calculation. In general, for added security, you should sign all the request headers that you plan to include in your request.
X-Amz-Signature	Provides the signature to authenticate your request. This signature must match the signature Amazon S3 calculates; otherwise, Amazon S3 denies the request. For example, 733255ef022bec3f2a8701cd61d4b371f3f28c9f193a1f02279211d48d5193d7

Calculating a Signature

The following diagram illustrates the signature calculation process.



The following table describes the functions that are shown in the diagram. You will need to implement code for these functions.

Description
Convert the string to lowercase.
Lowercase base 16 encoding.
Secure Hash Algorithm (SHA) cryptographic hash function.
Remove any leading or trailing whitespace.

	Description
	URI encode every byte. Uri-Encode() must enforce the following rules:

- URI encode every byte except the unreserved characters: 'A'-'Z', 'a'-'z', '0'-'9', '-', '.', '_', and '~'.
- The space character is a reserved character and must be encoded as "%20" (and not as "+").
- Each Uri-encoded byte is formed by a '%' and the two-digit hexadecimal value of the byte.
- Letters in the hexadecimal value must be uppercase, for example "%1A".
- Encode the forward slash character, '/', everywhere except in the object key name. For example, if the object key name is `photos/Jan/sample.jpg`, the forward slash in the key name is not encoded.

Caution
Standard Uri-Encode functions provided by your development platform may not work because of differences in implementation and related ambiguity in the underlying RFCs. We recommend that you write your own custom Uri-Encode function to ensure that your encoding will work.

The following is an example uri-encode() function in Java.

```
public static String uri-encode(CharSequence input, boolean encodeSlash) {
          StringBuilder result = new StringBuilder();
          for (int i = 0; i < input.length(); i++) {
              char ch = input.charAt(i);
              if ((ch >= 'A' && ch <= 'Z') || (ch >= 'a' && ch <= 'z') || (ch
 >= '0' && ch <= '9') || ch == '_' || ch == '-' || ch == '~' || ch == '.') {
                  result.append(ch);
              } else if (ch == '/') {
                  result.append(encodeSlash ? "%2F" : ch);
              } else {
                  result.append(toHexUTF8(ch));
              }
          }
          return result.toString();
      }
```

For more information about the signing process, see Authenticating Requests by Using the Authorization Header (Compute Checksum of the Entire Payload Prior to Transmission) - Signature Version 4 (p. 19). The process is generally the same except that the creation of **CanonicalRequest** in a presigned URL differs as follows:

- You don't include a payload hash in the **Canonical Request**, because when you create a presigned URL, you don't know anything about the payload. Instead, you use a constant string "**UNSIGNED-PAYLOAD**".
- The **Canonical Query String** must include all the query parameters from the preceding table except for `X-Amz-Signature`.
- **Canonical Headers** must include the HTTP `host` header. If you plan to include any of the `x-amz-*` headers, these headers must also be added for signature calculation. You can optionally add all other headers that you plan to include in your request. For added security, you should sign as many headers as possible.

An Example

Suppose you have an object `test.txt` in your `examplebucket` bucket. You want to share this object with others for a period of 24 hours (86400 seconds) by creating a presigned URL.

```
https://s3.amazonaws.com/examplebucket/test.txt
?X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20130524%2Fus-east-1%2Fs3%2Faws4_request
&X-Amz-Date=20130524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host
&X-Amz-Signature=<signature-value>
```

The following steps illustrate first the signature calculations and then construction of the presigned URL. The example makes the following additional assumptions:

- Request timestamp is `Fri, 24 May 2013 00:00:00 GMT`.
- The bucket is in the US Standard region, and the credential `Scope` and the `Signing Key` calculations use `us-east-1` as the region specifier. For more information, go to Regions and Endpoints in the *AWS General Reference*.

You can use this example as a test case to verify the signature that your code calculates; however, you must use the same bucket name, object key, time stamp, and the following example credentials:

Parameter	Value
AWSAccessKeyId	`AKIAIOSFODNN7EXAMPLE`
AWSSecretAccessKey	`wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`

1. **StringToSign**

   a. **CanonicalRequest**

```
GET
/test.txt
X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIOSFODNN7EX
AMPLE%2F20130524%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-
Date=20130524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host
host:examplebucket.s3.amazonaws.com

host
UNSIGNED-PAYLOAD
```

   b. **StringToSign**

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
3bfa292879f6447bbcda7001decf97f4a54dc650c8942174ae0a9121cf58ad04
```

2. **SigningKey**

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>","20130524"),"us-east-1"),"s3"),"aws4_request")
```

3. **Signature**

```
aeeed9bbccd4d02ee5c0109b86d86835f995330da4c265957d157751f604d404
```

The presigned URL that you can then test is as follows. Line feeds are added for readability:

```
https://examplebucket.s3.amazonaws.com/test.txt?X-Amz-Algorithm=AWS4-HMAC-
SHA256&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20130524%2Fus-east-
1%2Fs3%2Faws4_request&X-Amz-Date=20130524T000000Z&X-Amz-Expires=86400&X-Amz-
SignedHeaders=host&X-Amz-Signa
ture=aeeed9bbccd4d02ee5c0109b86d86835f995330da4c265957d157751f604d404
```

Examples: Signature Calculations in AWS Signature Version 4

Topics

- Signature Calculation Examples Using Java (AWS Signature Version 4) (p. 43)
- Examples of Signature Calculations Using C# (AWS Signature Version 4) (p. 44)

For authenticated requests, unless you are using the AWS SDKs, you have to write code to calculate signatures that provide authentication information in your requests. Signature calculation in AWS Signature Version 4 (see Authenticating Requests (AWS Signature Version 4) (p. 15)) can be a complex undertaking, and we recommend that you use the AWS SDKs whenever possible.

This section provides examples of signature calculations written in Java and C#. The code samples send the following requests and use the HTTP Authorization header to provide authentication information:

- PUT object – Separate examples illustrate both uploading the full payload at once and uploading the payload in chunks. For information about using the Authorization header for authentication, see Authenticating a Request in the Authorization Header (p. 17).
- GET object – This example generates a presigned URL to get an object. Query parameters provide the signature and other authentication information. Users can paste a presigned URL in their browser to retrieve the object, or you can use the URL to create a clickable link. For information about using query parameters for authentication, see Authenticating Requests by Using Query Parameters (AWS Signature Version 4) (p. 38).

The rest of this section describes the examples in Java and C#. The topics include instructions for downloading the samples and for executing them.

Signature Calculation Examples Using Java (AWS Signature Version 4)

The Java sample that shows signature calculation can be downloaded at https://s3.amazonaws.com/aws-java-sdk/samples/AWSS3SigV4JavaSamples.jar. In `RunAllSamples.java`, the `main()` function executes

sample requests to create an object, retrieve an object, and create a presigned URL for the object. The sample creates an object from the text string provided in the code:

```
PutS3ObjectSample.putS3Object(bucketName, regionName, awsAccessKey,
awsSecretKey);
GetS3ObjectSample.getS3Object(bucketName, regionName, awsAccessKey,
awsSecretKey);
PresignedUrlSample.getPresignedUrlToS3Object(bucketName, regionName, awsAccess
Key, awsSecretKey);
PutS3ObjectChunkedSample.putS3ObjectChunked(bucketName, regionName, awsAccessKey,
 awsSecretKey);
```

To test the examples on a Linux-based computer

The following instructions are tested on Linux operating system.

1. At a command prompt, change the directory to the directory that contains `AWSS3SigV4JavaSamples.jar`.
2. Extract the source files.

   ```
   jar xvf AWSS3SigV4JavaSamples.jar
   ```

3. In a text editor, open the file `./com/amazonaws/services/s3/samples/RunAllSamples.java`. Update code with the following information:

   - The name of a bucket where the new object can be created.

     Note
     The examples use a virtual-hosted style request to access the bucket. To avoid potential errors, ensure that your bucket name conforms to the bucket naming rules as explained in Bucket Restrictions and Limitations in the *Amazon Simple Storage Service Developer Guide*.

   - AWS region where the bucket resides.

     If bucket is in the US Standard region, use us-east-1 to specify the region. For a list of other AWS regions, go to Amazon Simple Storage Service (S3) in the *AWS General Reference*.

4. Compile the source code and store the compiled classes into the `bin/` directory.

   ```
   javac -d bin -source 6 -verbose com
   ```

5. Change the directory to `bin/`, and then execute `RunAllSamples`.

   ```
   java com.amazonaws.services.s3.sample.RunAllSamples
   ```

   The code runs all the methods in `main()`. For each request, the output will show the canonical request, the string to sign, and the signature.

Examples of Signature Calculations Using C# (AWS Signature Version 4)

The C# sample that shows signature calculation can be downloaded at http://docs.aws.amazon.com/AmazonS3/latest/API/samples/AmazonS3SigV4_Samples_CSharp.zip. In

`Program.cs`, the `main()` function executes sample requests to create an object, retrieve an object, and create a presigned URL for the object. The code for signature calculation is in the `\Signers` folder.

```
PutS3ObjectSample.Run(awsRegion, bucketName, "MySampleFile.txt");

Console.WriteLine("\n\n***********************************************");
PutS3ObjectChunkedSample.Run(awsRegion, bucketName, "MySampleFileChunked.txt");

Console.WriteLine("\n\n***********************************************");
GetS3ObjectSample.Run(awsRegion, bucketName, "MySampleFile.txt");

Console.WriteLine("\n\n***********************************************");
PresignedUrlSample.Run(awsRegion bucketName, "MySampleFile.txt");
```

To test the examples with Microsoft Visual Studio 2010 or later

1. Extract the .zip file.
2. Start Visual Studio, and then open the .sln file.
3. Update the code as follows:
   - In `Program.cs`, provide the bucket name and the AWS region where the bucket resides. The sample creates an object in this bucket.
4. Execute the code.
5. To verify that the object was created, copy the presigned URL that the program creates, and then paste it in a browser window.

Authenticating Requests in Browser-Based Uploads Using POST (AWS Signature Version 4)

Amazon S3 supports HTTP POST requests so that users can upload content directly to Amazon S3. Using HTTP POST to upload content simplifies uploads and reduces upload latency where users upload data to store in Amazon S3. This section describes how you authenticate HTTP POST requests. For more information about HTTP POST requests, how to create a form, create a POST policy, and an example, see Authenticating Requests in Browser-Based Uploads Using POST (AWS Signature Version 4) (p. 51).

To authenticate an HTTP POST request you do the following:

1. The form must include the following fields to provide signature and relevant information that Amazon S3 can use to re-calculate the signature upon receiving the request:

Element Name	Description
`policy`	The Base64 encoded security policy that describes what is permitted in the request. For signature calculation this policy is the string you sign. Amazon S3 must get this policy so it can re-calculate the signature.
`x-amz-algorithm`	The signing algorithm used. For AWS Signature Version 4, the value is `AWS4-HMAC-SHA256`.

Element Name	Description
x-amz-credential	In addition to your access key ID, this provides scope information you used in calculating the signing key for signature calculation. It is a string of the following form: <your-access-key-id>/<date>/<aws-region>/<aws-service>/aws4_request For example, AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request. . For Amazon S3, the *aws-service* string is "s3". For a list of aws-region strings, go to Regions and Endpoints in the *Amazon Web Services General Reference*.
x-amz-date	It is the date value in ISO8601 format. For example, 20130728T000000Z. It is the same date you used in creating the signing key. This must also be the same value you provide in the policy (x-amz-date) that you signed.
x-amz-signature	(AWS Signature Version 4) The HMAC-SHA256 hash of the security policy.

2. The POST policy must include the following elements:

Element Name	Description
x-amz-algorithm	The signing algorithm that you used to calculation the signature. For AWS Signature Version 4, the value is AWS4-HMAC-SHA256.
x-amz-credential	In addition to your access key ID, this provides scope information you used in calculating the signing key for signature calculation. It is a string of the following form: <your-access-key-id>/<date>/<aws-region>/<aws-service>/aws4_request For example, AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request. .
x-amz-date	The date value specified in the ISO8601 formatted string. For example, "20130728T000000Z". The date must be same that you used in creating the signing key for signature calculation.

3. For signature calculation the POST policy is the string to sign.

Calculating a Signature

The following diagram illustrates the signature calculation process.

To Calculate a signature

1. Create a policy using UTF-8 encoding.
2. Convert the UTF-8-encoded policy to Base64. The result is the string to sign.
3. Create the signature as an HMAC-SHA256 hash of the string to sign. You will provide the signing key as key to the hash function.
4. Encode the signature by using hex encoding.

For more information about creating HTML forms, security policies, and an example, see the following subtopics:

- Creating an HTML Form (Using AWS Signature Version 4) (p. 52)
- Creating a POST Policy (p. 56)
- Examples: Browser-Based Upload using HTTP POST (Using AWS Signature Version 4) (p. 61)
- Additional Considerations for Browser-Based Uploads (p. 64)

Amazon S3 Signature Version 4 Authentication Specific Policy Keys

The following table shows the policy keys related Amazon S3 Signature Version 4 authentication that can be in Amazon S3 policies. In a bucket policy, you can add these conditions to enforce specific behavior when requests are authenticated by using Signature Version 4. For example policies, see Bucket Policy Examples Using Signature Version 4 Related Condition Keys (p. 49).

Action	Applicable Keys	Description
`s3:*` or of the Amazon S3 actions.	`s3:signatureversion`	Identifies the version of AWS Signature that you want to support for authenticated requests. For authenticated requests, Amazon S3 supports both Signature Version 4 and Signature Version 2. You can add this condition in your bucket policy to require a specific signature version. Valid values: `"AWS"` identifies Signature Version 2 `"AWS4-HMAC-SHA256"` identifies Signature Version 4

	`s3:authType`	Amazon S3 supports various methods of authentication (see Authenticating Requests (AWS Signature Version 4) (p. 15). You can optionally use this condition key to restrict incoming requests to use a specific authentication method. For example, you can allow only the HTTP `Authorization` header to be used in request authentication.
		
		Valid values:
		
		`REST-HEADER`
		
		`REST-QUERY-STRING`
		
		`POST`

	`s3:signatureAge`	The length of time, in milliseconds, that a signature is valid in an authenticated request. In Signature Version 4, the signing key is valid for up to seven days (see Introduction to Signing Requests (p. 16). Therefore, the signatures are also valid for up to seven days. You can use this condition to further limit the signature age. Example value: `100`
	`s3:x-amz-content-sha256`	

Action	Applicable Keys	Description
		You can use this condition key to disallow unsigned content in your bucket. When you use Signature Version 4, for requests that use the `Authorization` header, you add the `x-amz-content-sha256` header in the signature calculation and then set its value to the hash payload. However, when you authenticate requests by using query parameters in a URL, you don't know the payload when you create the URL, and so you can't include the payload in the signature calculation. Instead, you set the `x-amz-content-sha256` header value to the constant string `UNSIGNED-PAYLOAD`. For more information, see Authenticating Requests by Using Query Parameters (AWS Signature Version 4) (p. 38). You can use this condition in your bucket policy to deny any uploads that use presigned URLs. Valid value: `UNSIGNED-PAYLOAD`

Bucket Policy Examples Using Signature Version 4 Related Condition Keys

Deny any Amazon S3 action on the `examplebucket` to anyone if request is authenticated using Signature Version 4.

```
{
   "Version": "2012-10-17",
   "Statement": [
         {
               "Sid": "Test",
               "Effect": "Deny",
               "Principal": "*",
               "Action": "s3:*",
               "Resource": "arn:aws:s3:::examplebucket/*",
               "Condition": {
                     "StringEquals": {
                           "s3:signatureversion": "AWS4-HMAC-SHA256"
                     }
               }
         }
   ]
}
```

The following bucket policy denies any Amazon S3 action on objects in `examplebucket` if signature is more than ten minutes old.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
```

```
         "Sid": "Deny request if signature is more than 10 min old",
         "Effect": "Deny",
         "Principal": "*",
         "Action": "s3:*",
         "Resource": "arn:aws:s3:::examplebucket3/*",
         "Condition": {
            "NumericGreaterThan": {
               "s3:signatureAge": 600000
            }
         }
      }
   ]
}
```

The following bucket policy allows only requests that use the Authorization header for request authentication. Any POST or presigned URL requests will be denied.

```
{
   "Version": "2012-10-17",
   "Statement": [
         {
               "Sid": "Allow only requests that use Authorization header for
request authentication. Deny POST or presigned URL requests.",
               "Effect": "Deny",
               "Principal": "*",
               "Action": "s3:*",
               "Resource": "arn:aws:s3:::examplebucket3/*",
               "Condition": {
                     "StringNotEquals": {
                           "s3:authType": "REST-HEADER"
                     }
               }
         }
   ]
}
```

The following bucket policy will deny any uploads that use presigned URLs.

```
{
   "Version": "2012-10-17",
   "Statement": [
         {
               "Sid": "Allow only requests that use Authorization header for
request authentication. Deny POST or presigned URL requests.",
               "Effect": "Deny",
               "Principal": "*",
               "Action": "s3:*",
               "Resource": "arn:aws:s3:::examplebucket3/*",
               "Condition": {
                     "StringNotEquals": {
                           "s3:x-amz-content-sha256": "UNSIGNED-PAYLOAD"
                     }
               }
         }
   ]
}
```

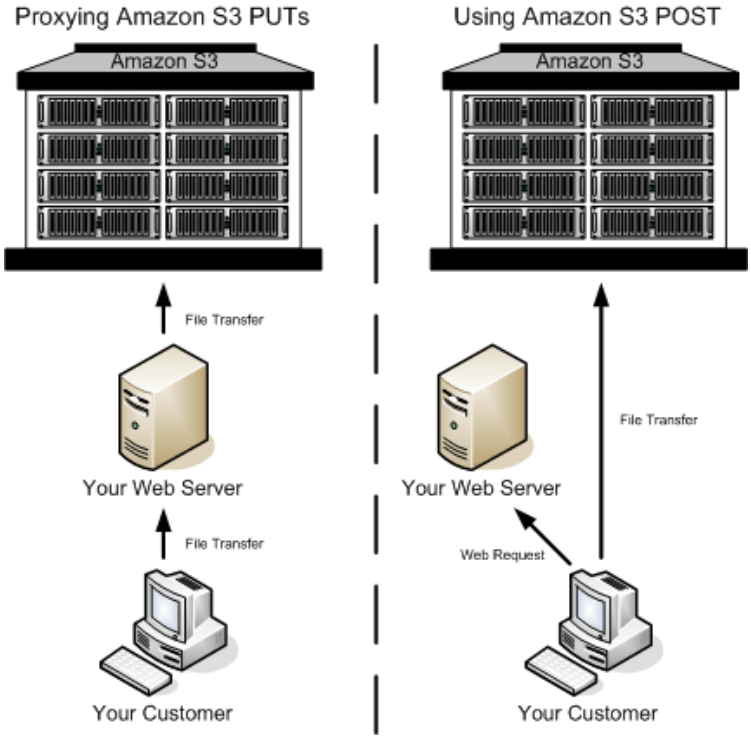
Authenticating Requests in Browser-Based Uploads Using POST (AWS Signature Version 4)

Topics

- Calculating a Signature (p. 52)
- Creating an HTML Form (Using AWS Signature Version 4) (p. 52)
- Creating a POST Policy (p. 56)
- Examples: Browser-Based Upload using HTTP POST (Using AWS Signature Version 4) (p. 61)
- Additional Considerations for Browser-Based Uploads (p. 64)

Amazon S3 supports HTTP POST requests so that users can upload content directly to Amazon S3. Using HTTP POST to upload content simplifies uploads and reduces upload latency where users upload data to store in Amazon S3 .

The following figure shows an Amazon S3 upload using a POST request.



Uploading Using POST

1	The user accesses your page from a web browser.
2	Your web page contains an HTTP form that contains all the information necessary for the user to upload content to Amazon S3.
3	The user uploads content to Amazon S3 through the web browser.

The process for sending browser-based POST requests is as follows:

1. Create an HTML form that your users can access in order to upload objects to your Amazon S3 bucket.
2. Create a security policy specifying conditions restricting what you want to allow in the request, such as bucket name where objects can be uploaded, key name prefixes that you want to allow for the object being created.
3. Create signature that is based on the policy. For authenticated requests, the form must include a valid signature and the policy.

The following section describes how to create a signature to authenticate a request. For information about creating forms and security policies, see Creating an HTML Form (Using AWS Signature Version 4) (p. 52).

Calculating a Signature

For authenticated requests, the HTML form must include fields for a security policy and a signature. A security policy (see Creating a POST Policy (p. 56)) controls what is allowed in the request. For signature calculation, the security policy is the string to sign (see Introduction to Signing Requests (p. 16)).

To Calculate a signature

1. Create a policy using UTF-8 encoding.
2. Convert the UTF-8-encoded policy to Base64. The result is the string to sign.
3. Create the signature as an HMAC-SHA256 hash of the string to sign. You will provide the signing key as key to the hash function.
4. Encode the signature by using Base64.

For more information about creating HTML forms, security policies, and an example, see the following subtopics:

- Creating an HTML Form (Using AWS Signature Version 4) (p. 52)
- Creating a POST Policy (p. 56)
- Examples: Browser-Based Upload using HTTP POST (Using AWS Signature Version 4) (p. 61)
- Additional Considerations for Browser-Based Uploads (p. 64)

Creating an HTML Form (Using AWS Signature Version 4)

Topics

- HTML Form Declaration (p. 53)
- HTML Form Fields (p. 53)

To allow users to upload content to Amazon S3 by using their browsers (HTTP POST requests), you use HTML forms. HTML forms consist of a form declaration and form fields. The form declaration contains high-level information about the request. The form fields contain detailed request information.

The form and policy must be UTF-8 encoded. You can apply UTF-8 encoding to the form by specifying it in the HTML heading or as a request header. The following is an example of UTF-8 encoding in the HTML heading.

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
```

Following is an example of UTF-8 encoding in a request header.

```
Content-Type: text/html; charset=UTF-8
```

Note
The form data and boundaries (excluding the contents of the file) cannot exceed 20K.

HTML Form Declaration

The HTML declaration in a form has the following three attributes:

- **action —** The URL that processes the request, which must be set to the URL of the bucket. For example, if the name of your bucket is "examplebucket", the URL is "http://examplebucket.s3.amazonaws.com/".

  Note
  The key name is specified in a form field.
- **method —** The method must be POST.
- **enclosure type —** The enclosure type (enctype) must be set to multipart/form-data for both file uploads and text area uploads. For more information about enctype, go to RFC 1867.

This is a form declaration for the bucket "examplebucket".

```
<form action="http://examplebucket.s3.amazonaws.com/" method="post"

enctype="multipart/form-data">
```

HTML Form Fields

The following table describes a list of fields that can be used within a form. Among other fields, there is a signature field that you can use to authenticate requests. There are fields for you to specify the signature calculation algorithm (x-amz-algorithm), the credential scope (x-amz-credential) that you used to generate the signing key, and the date (x-amz-date) used to calculate signature. Amazon S3 uses this information to re-create the signature. If the signatures match, Amazon S3 processes the request.

Note
The variable `${filename}` is automatically replaced with the name of the file provided by the user and is recognized by all form fields. If the browser or client provides a full or partial path to the file, only the text following the last slash (/) or backslash (\) will be used (e.g., "`C:\Program`

`Files\directory1\file.txt`" will be interpreted as "`file.txt`"). If no file or file name is provided, the variable is replaced with an empty string.

The elements in the table that are marked conditional are required for authenticated requests. If you don't provide these elements, such as the `policy` element, the request is assumed to be anonymous and will succeed only if you have configured the bucket for public read and write.

Element Name	Description	Required
`acl`	An Amazon S3 access control list. If an invalid access control list is specified, Amazon S3 denies the request. For more information on ACLs, see Using Amazon S3 ACLs. Type: String Default: private Valid Values: `private | public-read | public-read-write | authenticated-read | bucket-owner-read | bucket-owner-full-control`	No
`Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires`	REST-specific headers. For more information, see PUT Object (p. 250).	No
`key`	The key name of the uploaded object. To use the file name provided by the user, use the ${filename} variable. For example, if a user `User1` uploads a file photo1.jpg and you specify `/user/user1/${filename}`, the file is stored as `/user/user1/photo1.jpg`. For more information, go to Object Key and Metadata in the *Amazon Simple Storage Service Developer Guide*.	Yes
`policy`	The Base64 encoded security policy that describes what is permitted in the request. For authenticated requests a policy is required. For anonymous requests the policy and the signature are optional. Requests without a security policy are considered anonymous and will succeed only on a publicly writable bucket.	Conditional
`success_action_redirect`	The URL to which the client is redirected upon successful upload. If `success_action_redirect` is not specified, or Amazon S3 cannot interpret the URL, Amazon S3 returns the empty document type that is specified in the `success_action_status` field. If the upload fails, Amazon S3 returns an error and does not redirect the user to another URL.	No

Element Name	Description	Required
success_action_status	The status code returned to the client upon successful upload if `success_action_redirect` is not specified. Valid values are `200`, `201`, or `204` (default). If the value is set to 200 or 204, Amazon S3 returns an empty document with the specified status code. If the value is set to 201, Amazon S3 returns an XML document with a 201 status code. For information on the content of the XML document, see POST Object (p. 238). If the value is not set or is invalid, Amazon S3 returns an empty document with a 204 status code. **Note** Some versions of the Adobe Flash player do not properly handle HTTP responses with an empty body. To support uploads through Adobe Flash, we recommend setting *success_action_status* to 201.	No
x-amz-algorithm	The signing algorithm used to authenticate the request. For AWS Signature Version 4, the value is `AWS4-HMAC-SHA256`. This field is required if a policy document is included with the request.	Conditional
x-amz-credential	In addition to your access key ID, this field also provides scope information identifying region and service for which the signature is valid. This should be the same scope you used in calculating the signing key for signature calculation. It is a string of the following form: <*your-access-key-id*>/<*date*>/<*aws-region*>/<*aws-service*>/aws4_request For example, AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request. . For Amazon S3, the *aws-service* string is "s3". For a list of aws-region strings, go to Regions and Endpoints in the *Amazon Web Services General Reference*. This is required if a policy document is included with the request.	Conditional
x-amz-date	It is the date value in ISO8601 format. For example, `20130728T000000Z`. It is the same date you used in creating the signing key. This must also be the same value you provide in the policy (`x-amz-date`) that you signed. This is required if a policy document is included with the request.	Conditional

Element Name	Description	Required
x-amz-security-token	A security token used by Amazon DevPay and session credentials If the request is using Amazon DevPay then it requires two `x-amz-security-token` form fields: one for the product token and one for the user token. For more information, go to Using DevPay. If the request is using session credentials, then it requires one `x-amz-security-token` form . For more information, go to Welcome in the *AWS Security Token Service* guide.	No
x-amz-signature	(AWS Signature Version 4) The HMAC-SHA256 hash of the security policy. This field is required if a policy document is included with the request.	Conditional
Other field names prefixed with x-amz-meta-	User-specified metadata. Amazon S3 does not validate or use this data. For more information, see PUT Object (p. 250).	No
file	File or text content. The file or content must be the last field in the form. You cannot upload more than one file at a time.	Yes

Now that you know how to create forms, next you can create security policy that you can sign. For more information, see Creating a POST Policy (p. 56).

Creating a POST Policy

Topics

- Expiration (p. 57)
- Conditions (p. 57)
- Condition Matching (p. 59)
- Character Escaping (p. 59)

The policy required for making authenticated requests using HTTP POST is a UTF-8 and Base64 encoded document written in JavaScript Object Notation (JSON) that specifies conditions that the request must meet. Depending on how you design your policy document, you can control per-upload, per-user, for all uploads, or according to other designs that meet your needs.

> **Note**
> Although the policy document is optional, we highly recommend that you use one in order to control what is allowed in the request. If you make the bucket publicly writable, you have no control at all over what users might write to your bucket.

The following is an example of a POST policy document.

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"acl": "public-read" },
    {"bucket": "johnsmith" },
    ["starts-with", "$key", "user/eric/"],
  ]
}
```

The POST policy always contain the `expiration` and `conditions` elements.

Expiration

The `expiration` element specifies the expiration date and time of the POST policy in ISO8601 GMT date format. For example, "2013-08-01T12:00:00.000Z" specifies that the POST policy is not valid after midnight GMT on August 1, 2013.

Conditions

The `conditions` in a POST policy is an array of objects, each of which is used to validate the contents of the uploaded object. You can use these conditions to restrict what is allowed in the request. Each form field that you specify in a form (except `x-amz-signature`, `file`, `policy`, and field names that have an `x-ignore-` prefix) must appear in the list of conditions.

> **Note**
> All variables within the form are expanded prior to validating the POST policy. Therefore, all condition matching should be against the expanded form fields. For example, if you set the key form field to `user/user1/${filename}`, your POST policy might be `[ "starts-with", "$key", "user/user1/" ]`. Do not enter `[ "starts-with", "$key", "user/user1/${filename}" ]`. For more information, see Condition Matching (p. 59).

Policy document conditions are described in the following table.

Element Name	Description
acl	Specifies conditions the ACL specified in the form must meet. This condition supports exact matching and *`starts-with`* condition match type discussed in the following section.
content-length-range	The minimum and maximum allowable size for the uploaded content. This condition supports `content-length-range` condition match type.
Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires	REST-specific headers. For more information, see POST Object (p. 238). This condition supports exact matching and *`starts-with`* condition match type.
key	The key name of the uploaded object. This condition supports exact matching and *`starts-with`* condition match type.
success_action_redirect, redirect	The URL to which the client is redirected upon successful upload. This condition supports exact matching and *`starts-with`* condition match type.

Element Name	Description
success_action_status	The status code returned to the client upon successful upload if success_action_redirect is not specified. This condition supports exact matching.
x-amz-algorithm	The signing algorithm that you used to calculation the signature. For AWS Signature Version 4, the value is `AWS4-HMAC-SHA256`. This condition supports exact matching.
x-amz-credential	The credentials that you used to calculate the signature. It provides access key ID and scope information identifying region and service for which the signature is valid. This should be the same scope you used in calculating the signing key for signature calculation. It is a string of the following form: *<your-access-key-id>*/*<date>*/*<aws-region>*/*<aws-service>*/aws4_request For example, `AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request`. For Amazon S3, the aws-service string is "s3". For a list of aws-region strings, go to Regions and Endpoints in the *Amazon Web Services General Reference*. This is required if a POST policy document is included with the request. This condition supports exact matching.
x-amz-date	The date value specified in the ISO8601 formatted string. For example, "20130728T000000Z". The date must be same that you used in creating the signing key for signature calculation. This is required if a POST policy document is included with the request. This condition supports exact matching.
x-amz-security-token	Amazon DevPay security token. Each request that uses Amazon DevPay requires two `x-amz-security-token` form fields: one for the product token and one for the user token. As a result, the values must be separated by commas. For example, if the user token is *`eW91dHViZQ==`* and the product token is *`b0hnNVNKWVJIQTA=`*, you set the POST policy entry to: `{ "x-amz-security-token": "eW91dHViZQ==,b0hnNVNKWVJIQTA=" }`. For more information about Amazon DevPay, see Using DevPay.
Other form field names prefixed with x-amz-meta-	User-specified metadata. This condition supports exact matching and *`starts-with`* condition match type.

Note
If your toolkit adds additional form fields (e.g., Flash adds filename), you must add them to the POST policy document. If you can control this functionality, prefix `x-ignore-` to the field so Amazon S3 ignores the feature and it won't affect future versions of this feature.

Condition Matching

Following is a table that describes condition matching types. Although you must specify one condition for each form field that you specify in the form, you can create more complex matching criteria by specifying multiple conditions for a form field.

Condition Match Type	Description

Exact Matches	The form field value must match the value specified. This example indicates that the ACL must be set to public-read:
	`{"acl": "public-read" }`
	This example is an alternate way to indicate that the ACL must be set to public-read:
	`[ "eq", "$acl", "public-read" ]`

Starts With	The value must start with the specified value. This example indicates that the key must start with user/user1: `["starts-with", "$key", "user/user1/"]`
Matching Any Content	To configure the POST policy to allow any content within a form field, use `starts-with` with an empty value (""). This example allows any value for `success_action_redirect`: `["starts-with", "$success_action_redirect", ""]`
Specifying Ranges	For form fields that accept a range, separate the upper and lower limit with a comma. This example allows a file size from 1 to 10 MiB: `["content-length-range", 1048579, 10485760]`

Character Escaping

Characters that must be escaped within a POST policy document are described in the following table.

Escape Sequence	Description
\\	Backslash
\$	Dollar symbol
\b	Backspace
\f	Form feed

Escape Sequence	Description
\n	New line
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab
\u*xxxx*	All Unicode characters

Now that you are acquainted with forms and policies, you can see try an upload example. You will need to write the code to calculate the signature. The example provides a sample form, and a POST policy. For more information, see Examples: Browser-Based Upload using HTTP POST (Using AWS Signature Version 4) (p. 61).

Examples: Browser-Based Upload using HTTP POST (Using AWS Signature Version 4)

Topics

- File Upload (p. 61)

File Upload

This example provides a sample POST policy and a form that you can use to upload a file attachment. You can use this date to calculate signature and test the application. The example uses following the following example access keys in the signature calculations.

Parameter	Value
AWSAccessKeyId	`AKIAIOSFODNN7EXAMPLE`
AWSSecretAccessKey	`wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`

Sample Policy and Form

The following POST policy supports uploads to Amazon S3 with specific conditions.

```
{ "expiration": "2013-08-06T12:00:00.000Z",
  "conditions": [
    {"bucket": "examplebucket"},
    ["starts-with", "$key", "user/user1/"],
    {"acl": "public-read"},
    {"success_action_redirect": "http://acl6.s3.amazonaws.com/successful_up
load.html"},
    ["starts-with", "$Content-Type", "image/"],
    {"x-amz-meta-uuid": "14365123651274"},
    ["starts-with", "$x-amz-meta-tag", ""],

    {"x-amz-credential": "AKIAIOSFODNN7EXAMPLE/20130806/us-east-1/s3/aws4_re
quest"},
    {"x-amz-algorithm": "AWS4-HMAC-SHA256"},
    {"x-amz-date": "20130806T000000Z" }
  ]
}
```

This POST policy sets the following conditions on the request:

- The upload must occur before midnight UTC on August 6, 2013.
- The content can be uploaded only to the `examplebucket`. The bucket must be in the region that you specified in the credential scope (`x-amz-credential` form parameter), because the signature you provided is valid only within this scope.
- You can provide any key name that starts with `user/user1`. For example, `user/user1/MyPhoto.jpg`.
- The ACL must be set to `public-read`
- If the upload succeeds, the user's browser is redirected to `http://examplebucket.s3.amazonaws.com/successful_upload.html`
- The object must be an image file.

- The `x-amz-meta-uuid` tag must be set to `14365123651274`
- The `x-amz-meta-tag` can contain any value

The following is a Base64 encoded version of this POST policy.

```
eyAiZXhwaXJhdGlvbiI6ICIyMDEzLTA4LTA3VDEyOjAwOjAwLjAwMFoiLA0KI
CAiY29uZGl0aW9ucyI6IFsNCiAgICB7ImJ1Y2tldCI6ICJleGFtcGxlYnVja2V0In0sDQogICAg
WyJzdGFydHMtd2l0aCIsICIka2V5IiwgInVzZXIvdXNlcjEvIl0sDQogICAgeyJhY2wiOiAicHVi
bGljLXJlYWQifSwNCiAgICB7InN1Y2Nlc3NfYWN0aW9uX3JlZGlyZWN0Ijo
gImh0dHA6Ly9leGFtcGxlYnVja2V0LnMzLmFtYXpvbmF3cy5jb20vc3VjY2Vzc2Z1bF91cGxvYWQua
HRtbCJ9LA0KICAgIFsic3RhcnRzLXdpdGgiLCAiJENvbnRlbnQtVHlwZSIsICJpbWFnZS8iXSwN
CiAgICB7IngtYW16LW1ldGEtdXVpZCI6ICIxNDM2NTEyMzY1MTI3NCJ9LA0KICAgIFsic3RhcnRzLXd
pdGgiLCAiJHgtYW16LW1ldGEtdGFnIiwgIiJdLA0KDQogICAgeyJ4LWFtei1jcmVkZW50aWFsIjo
gIkFLSUFJT1NGT0ROTjdFWEFNUExFLzIwMTMwODA2L3VzLWVhc3QtMS9zMy9hd3M0X3JlcXVlc3Qif
SwNCiAgICB7IngtYW16LWFsZ29yaXRobSI6ICJBV1M0LUhNQUMtU0hBMjU2In0sDQogICAgeyJ4LW
Ftei1kYXRlIjogIjIwMTMwODA2VDAwMDAwMFoiIH0NCiAgXQ0KfQ==
```

Using example credentials to create a signature, the signature value is as follows:

```
21496b44de44ccb73d545f1a995c68214c9cb0d41c45a17a5daeec0b1a6db047
```

The following example form specifies the preceding POST policy and supports a POST request to the `examplebucket`. Copy/paste the content in a text editor and save it as exampleform.html. You can then upload image files to the specific bucket. You request will succeed if you signature you provide matches the signature Amazon S3 calculates.

```
<html>
  <head>

    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

  </head>
  <body>

  <form action="http://examplebucket.s3.amazonaws.com/" method="post" enc
type="multipart/form-data">
    Key to upload:
    <input type="input"  name="key" value="user/user1/" /><br />
    <input type="hidden" name="acl" value="public-read" />
   <input type="hidden" name="success_action_redirect" value="http://examplebuck
et.s3.amazonaws.com/successful_upload.html" />
    Content-Type:
    <input type="input"  name="Content-Type" value="image/jpeg" /><br />
    <input type="hidden" name="x-amz-meta-uuid" value="14365123651274" />
    <input type="text"   name="X-Amz-Credential" value="AKIAIOSFODNN7EX
AMPLE/20130806/us-east-1/s3/aws4_request" />
    <input type="text"   name="X-Amz-Algorithm" value="AWS4-HMAC-SHA256" />
    <input type="text"   name="X-Amz-Date" value="20130806T000000Z" />

    Tags for File:
    <input type="input"  name="x-amz-meta-tag" value="" /><br />
    <input type="hidden" name="Policy" value='<Base64-encoded policy string>'
/>
    <input type="hidden" name="X-Amz-Signature" value="<signature-value>" />
```

```
    File:
    <input type="file"   name="file" /> <br />
    <!-- The elements after this will be ignored -->
    <input type="submit" name="submit" value="Upload to Amazon S3" />
  </form>

</html>
```

Sample Request

The following is a sample POST request, it does not include object data being uploaded.

```
POST http://examplebucket.s3.amazonaws.com/ HTTP/1.1
Host: examplebucket.s3.amazonaws.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:22.0) Gecko/20100101 Fire
fox/22.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: keep-alive
Content-Type: multipart/form-data; boundary=---------------------------
195571638125373
Content-Length: 5680

-----------------------------195571638125373
Content-Disposition: form-data; name="key"

user/user1/
-----------------------------195571638125373
Content-Disposition: form-data; name="acl"

public-read
-----------------------------195571638125373
Content-Disposition: form-data; name="success_action_redirect"

http://examplebucket.s3.amazonaws.com/successful_upload.html
-----------------------------195571638125373
Content-Disposition: form-data; name="Content-Type"

image/jpeg
-----------------------------195571638125373
Content-Disposition: form-data; name="x-amz-meta-uuid"

14365123651274
-----------------------------195571638125373
Content-Disposition: form-data; name="X-Amz-Credential"

AKIAIOSFODNN7EXAMPLE/20130806/us-east-1/s3/aws4_request
-----------------------------195571638125373
Content-Disposition: form-data; name="X-Amz-Algorithm"

AWS4-HMAC-SHA256
-----------------------------195571638125373
Content-Disposition: form-data; name="X-Amz-Date"

20130806T000000Z
```

```
----------------------------195571638125373
Content-Disposition: form-data; name="x-amz-meta-tag"


----------------------------195571638125373
Content-Disposition: form-data; name="Policy"

eyAiZXhwaXJhdGlvbiI6ICIyMDEzLTA4LTA3VDEyOjAwOjAwLjAwMFoiLA0KI
CAiY29uZGl0aW9ucyI6IFsNCiAgICB7ImJ1Y2tldCI6ICJleGFtcGxlYnVja2V0In0sDQogICAg
WyJzdGFydHMtd2l0aCIsICIka2V5IiwgInVzZXIvdXNlcjEvIl0sDQogICAgeyJhY2wiOiAicHVi
bGljLXJlYWQifSwNCiAgICB7InN1Y2Nlc3NfYWN0aW9uX3JlZGlyZWN0Ijo
gImh0dHA6Ly9leGFtcGxlYnVja2V0LnMzLmFtYXpvbmF3cy5jb20vc3VjY2Vzc2Z1bF91cGxvYWQua
HRtbCJ9LA0KICAgIFsic3RhcnRzLXdpdGgiLCAiJENvbnRlbnQtVHlwZSIsICJpbWFnZS8iXSwN
CiAgICB7IngtYW16LW1ldGEtdXVpZCI6ICIxNDM2NTEyMzY1MTI3NCJ9LA0KICAgIFsic3RhcnRzLXd
pdGgiLCAiJHgtYW16LW1ldGEtdGFnIiwgIiJdLA0KDQogICAgeyJ4LWFtei1jcmVkZW50aWFsIjo
gIkFLSUFJT1NGT0ROTjdFWEFNUExFLzIwMTMwODA2L3VzLWVhc3QtMS9zMy9hd3M0X3JlcXVlc3Qif
SwNCiAgICB7InggtYW16LWFsZ29yaXRobSI6ICJBV1M0LUhNQUMtU0hBMjU2In0sDQogICAgeyJ4LW
Ftei1kYXRlIjogIjIwMTMwODA2VDAwMDAwMFoiIH0NCiAgXQ0KfQ==
----------------------------195571638125373
Content-Disposition: form-data; name="X-Amz-Signature"

21496b44de44ccb73d545f1a995c68214c9cb0d41c45a17a5daeec0b1a6db047
----------------------------195571638125373
Content-Disposition: form-data; name="file"; filename="HappyFace.jpg"
Content-Type: image/jpeg
<content body not shown>
```

Additional Considerations for Browser-Based Uploads

This section discusses additional considerations for uploading objects with an HTTP POST request.

POST with Adobe Flash

This section describes how to use POST with Adobe Flash.

Adobe Flash Player Security

By default, the Adobe Flash Player security model prohibits making network connections to servers outside the domain that serves the Adobe Flash (.swf) file.

To override the default, you must upload a publicly readable crossdomain.xml file to the bucket that will accept POST uploads. Here is a sample crossdomain.xml file:

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM
"http://www.macromedia.com/xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
  <allow-access-from domain="*" secure="false" />
</cross-domain-policy>
```

For more information about the Adobe Flash security model, go to the Adobe web site.

When you add the crossdomain.xml file to your bucket, any Adobe Flash Player can connect to the crossdomain.xml file within your bucket; however, crossdomain.xml does not grant access to the Amazon S3 bucket.

Other Adobe Flash Considerations

The FileReference class in the Adobe Flash API adds the *Filename* form field to the POST request. When you build an Adobe Flash application that uploads files to Amazon S3 by using the FileReference class, include the following condition in your policy:

```
['starts-with', '$Filename', '']
```

Some versions of the Adobe Flash Player do not properly handle HTTP responses that have an empty body. To configure POST to return a response that does not have an empty body, set *success_action_status* to 201. Amazon S3 will then return an XML document with a 201 status code. For information about his is an optional element, and currently the only allowed value is the content of the XML document, see POST Object (p. 238). For information on form fields, see HTML Form Fields (p. 53).

Operations on the Service

Topics

- GET Service (p. 65)

This section describes operations you can perform on the Amazon S3 service.

GET Service

Description

This implementation of the GET operation returns a list of all buckets owned by the authenticated sender of the request.

To authenticate a request, you must use a valid AWS Access Key ID that is registered with Amazon S3. Anonymous requests cannot list buckets, and you cannot list buckets that you did not create.

Requests

Syntax

```
GET / HTTP/1.1
Host: s3.amazonaws.com
Date: date
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))
```

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

This implementation of the operation uses only request headers that are common to all operations. For more information, see Common Request Headers (p. 12).

Request Elements

This implementation of the operation does not use request elements.

Responses

Response Elements

Name	Description
Bucket	Container for bucket information. Type: Container Children: Name, CreationDate Ancestor: ListAllMyBucketsResult.Buckets
Buckets	Container for one or more buckets. Type: Container Children: Bucket Ancestor: ListAllMyBucketsResult
CreationDate	Date the bucket was created. Type: date (of the form yyyy-mm-ddThh:mm:ss.timezone, e.g., 2009-02-03T16:45:09.000Z) Ancestor: ListAllMyBucketsResult.Buckets.Bucket
DisplayName	Bucket owner's display name. Type: String Ancestor: ListAllMyBucketsResult.Owner
ID	Bucket owner's user ID. Type: String Ancestor: ListAllMyBucketsResult.Owner
ListAllMyBucketsResult	Container for response. Type: Container Children: Owner, Buckets Ancestor: None
Name	Bucket's name. Type: String Ancestor: ListAllMyBucketsResult.Buckets.Bucket
Owner	Container for bucket owner information. Type: Container Ancestor: ListAllMyBucketsResult

Special Errors

This implementation of the operation does not return special errors. For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

Sample Request

The GET operation on the Service endpoint (s3.amazonaws.com) returns a list of all of the buckets owned by the authenticated sender of the request.

```
GET / HTTP/1.1
Host: s3.amazonaws.com
Date: Wed, 01 Mar  2006 12:00:00 GMT
Authorization: authorization string
```

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<ListAllMyBucketsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01"
  <Owner>
    <ID>bcaf1ffd86f461ca5fb16fd081034f</ID>
    <DisplayName>webfile</DisplayName>
  </Owner>
  <Buckets>
    <Bucket>
      <Name>quotes</Name>
      <CreationDate>2006-02-03T16:45:09.000Z</CreationDate>
    </Bucket>
    <Bucket>
      <Name>samples</Name>
      <CreationDate>2006-02-03T16:41:58.000Z</CreationDate>
    </Bucket>
  </Buckets>
</ListAllMyBucketsResult>
```

Related Resources

- GET Bucket (List Objects) (p. 81)
- GET Object (p. 212)

Operations on Buckets

Topics

- DELETE Bucket (p. 69)
- DELETE Bucket cors (p. 71)
- DELETE Bucket lifecycle (p. 73)
- DELETE Bucket policy (p. 75)
- DELETE Bucket tagging (p. 77)
- DELETE Bucket website (p. 79)
- GET Bucket (List Objects) (p. 81)
- GET Bucket acl (p. 89)
- GET Bucket cors (p. 92)
- GET Bucket lifecycle (p. 95)
- GET Bucket policy (p. 101)
- GET Bucket location (p. 103)
- GET Bucket logging (p. 105)
- GET Bucket notification (p. 108)
- GET Bucket tagging (p. 111)
- GET Bucket Object versions (p. 114)
- GET Bucket requestPayment (p. 126)
- GET Bucket versioning (p. 128)
- GET Bucket website (p. 131)
- HEAD Bucket (p. 133)
- List Multipart Uploads (p. 135)
- PUT Bucket (p. 144)
- PUT Bucket acl (p. 150)
- PUT Bucket cors (p. 157)
- PUT Bucket lifecycle (p. 162)
- PUT Bucket policy (p. 171)
- PUT Bucket logging (p. 173)
- PUT Bucket notification (p. 178)
- PUT Bucket tagging (p. 182)
- PUT Bucket requestPayment (p. 185)
- PUT Bucket versioning (p. 187)
- PUT Bucket website (p. 191)

This section describes operations you can perform on Amazon S3 buckets.

DELETE Bucket

Description

This implementation of the DELETE operation deletes the bucket named in the URI. All objects (including all object versions and delete markers) in the bucket must be deleted before the bucket itself can be deleted.

Requests

Syntax

```
DELETE / HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))
```

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

This implementation of the operation uses only request headers that are common to all operations. For more information, see Common Request Headers (p. 12).

Request Elements

This implementation of the operation does not use request elements.

Responses

Response Headers

This implementation of the operation uses only response headers that are common to most responses. For more information, see Common Response Headers (p. 14).

Response Elements

This implementation of the operation does not return response elements.

Special Errors

This implementation of the operation does not return special errors. For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

Sample Request

This request deletes the bucket named "quotes".

```
DELETE / HTTP/1.1
Host: quotes.s3.amazonaws.com
Date: Wed, 01 Mar  2006 12:00:00 GMT
Authorization: authorization string
```

Sample Response

```
HTTP/1.1 204 No Content
x-amz-id-2: JuKZqmXuiwFeDQxhD7M8KtsKobSzWA1QEjLbTMTagkKdBX2z7Il/jGhDeJ3j6s80
x-amz-request-id: 32FE2CEB32F5EE25
Date: Wed, 01 Mar  2006 12:00:00 GMT
Connection: close
Server: AmazonS3
```

Related Resources

- PUT Bucket (p. 144)
- DELETE Object (p. 200)

DELETE Bucket cors

Description

Deletes the *cors* configuration information set for the bucket.

To use this operation, you must have permission to perform the `s3:PutCORSConfiguration` action. The bucket owner has this permission by default and can grant this permission to others.

For information more about `cors`, go to Enabling Cross-Origin Resource Sharing in the *Amazon Simple Storage Service Developer Guide*.

Requests

Syntax

```
DELETE /?cors HTTP/1.1
Host: bucketname.s3.amazonaws.com
Date: date
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))
```

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

This implementation of the operation uses only request headers that are common to all operations. For more information, see Common Request Headers (p. 12).

Request Elements

This implementation of the operation does not use request elements.

Responses

Response Headers

This implementation of the operation uses only response headers that are common to most responses. For more information, see Common Response Headers (p. 14).

Examples

Example 1: Retrieve cors subresource

The following DELETE request deletes the `cors` subresource from the specified bucket. This action removes `cors` configuration that is stored in the subresource.

Sample Request

```
DELETE /?cors HTTP/1.1
Host: examplebucket.s3.amazonaws.com
```

```
Date: Tue, 13 Dec 2011 19:14:42 GMT
Authorization: signatureValue
```

Sample Response

```
HTTP/1.1 204 No Content
x-amz-id-2: 0FmFIWsh/PpBuzZ0JFRC55ZGVmQW4SHJ7xVDqKwhEdJmf3q63RtrvH8ZuxW1Bol5
x-amz-request-id: 0CF038E9BCF63097
Date: Tue, 13 Dec 2011 19:14:42 GMT
Server: AmazonS3
Content-Length: 0
```

Related Resources

- PUT Bucket cors (p. 157)
- DELETE Bucket cors (p. 71)
- OPTIONS object (p. 235)

DELETE Bucket lifecycle

Description

Deletes the lifecycle configuration from the specified bucket. Amazon S3 removes all the lifecycle configuration rules in the lifecycle subresource associated with the bucket. Your objects never expire, and Amazon S3 no longer automatically deletes any objects on the basis of rules contained in the deleted lifecycle configuration.

To use this operation, you must have permission to perform the `s3:PutLifecycleConfiguration` action. By default, the bucket owner has this permission and the bucket owner can grant this permission to others.

There is usually some time lag before lifecycle configuration deletion is fully propagated to all the Amazon S3 systems.

For more information about the object expiration, go to Object Expiration in the *Amazon Simple Storage Service Developer Guide*.

Requests

Syntax

```
DELETE /?lifecycle HTTP/1.1
Host: bucketname.s3.amazonaws.com
Date: date
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))
```

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

This implementation of the operation uses only request headers that are common to all operations. For more information, see Common Request Headers (p. 12).

Request Elements

This implementation of the operation does not use request elements.

Responses

Response Headers

This implementation of the operation uses only response headers that are common to most responses. For more information, see Common Response Headers (p. 14).

Examples

Sample Request

The following DELETE request deletes the `lifecycle` subresource from the specified bucket. This removes lifecycle configuration stored in the subresource.

```
DELETE /?lifecycle HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Date: Wed, 14 Dec 2011 05:37:16 GMT
Authorization: signatureValue
```

Sample Response

The following successful response shows Amazon S3 returning a `204 No Content` response. Objects in your bucket no longer expire.

```
HTTP/1.1 204 No Content
x-amz-id-2: Uuag1LuByRx9e6j5OnimrSAMPLEtRPfTaOAa==
x-amz-request-id: 656c76696e672SAMPLE5657374
Date: Wed, 14 Dec 2011 05:37:16 GMT
Connection: keep-alive
Server: AmazonS3
```

Related Resources

- PUT Bucket lifecycle (p. 162)
- GET Bucket lifecycle (p. 95)

DELETE Bucket policy

Description

This implementation of the `DELETE` operation uses the *`policy`* subresource to delete the policy on a specified bucket. To use the operation, you must have *`DeletePolicy`* permissions on the specified bucket and be the bucket owner.

If you do not have *`DeletePolicy`* permissions, Amazon S3 returns a `403 Access Denied` error. If you have the correct permissions, but are not the bucket owner , Amazon S3 returns a `405 Method Not Allowed` error. If the bucket doesn't have a policy, Amazon S3 returns a `204 No Content` error. There are restrictions about who can create bucket policies and which objects in a bucket they can apply to. For more information, go to Using Bucket Policies.

Requests

Syntax

```
DELETE /?policy HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))
```

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

This implementation of the operation uses only request headers that are common to all operations. For more information, see Common Request Headers (p. 12).

Request Elements

This implementation of the operation does not use request elements.

Responses

Response Headers

This implementation of the operation uses only response headers that are common to most responses. For more information, see Common Response Headers (p. 14).

Response Elements

The response elements contain the status of the `DELETE` operation including the error code if the request failed.

Special Errors

This implementation of the operation does not return special errors. For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

Sample Request

This request deletes the bucket named `BucketName`.

```
DELETE /?policy HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: Tue, 04 Apr 2010 20:34:56 GMT
Authorization: signatureValue
```

Sample Response

```
HTTP/1.1 204 No Content
x-amz-id-2: Uuag1LuByRx9e6j5OnimrSAMPLEtRPfTaOFg==
x-amz-request-id: 656c76696e672SAMPLE5657374
Date: Tue, 04 Apr 2010 20:34:56 GMT
Connection: keep-alive
Server: AmazonS3
```

Related Resources

- PUT Bucket (p. 144)
- DELETE Object (p. 200)

DELETE Bucket tagging

Description

This implementation of the `DELETE` operation uses the *`tagging`* subresource to remove a tag set from the specified bucket.

To use this operation, you must have permission to perform the `s3:PutBucketTagging` action. By default, the bucket owner has this permission and can grant this permission to others.

Requests

Syntax

```
DELETE /?tagging HTTP/1.1
Host: bucketname.s3.amazonaws.com
Date: date
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))
```

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

This implementation of the operation uses only request headers that are common to all operations. For more information, see Common Request Headers (p. 12).

Request Elements

This implementation of the operation does not use request elements.

Responses

Response Headers

This implementation of the operation uses only response headers that are common to most responses. For more information, see Common Response Headers (p. 14).

Examples

Sample Request

The following DELETE request deletes the tag set from the specified bucket.

```
DELETE /?tagging HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Date: Wed, 14 Dec 2011 05:37:16 GMT
Authorization: signatureValue
```

Sample Response

The following successful response shows Amazon S3 returning a `204 No Content` response. The tag set for the bucket has been removed.

```
HTTP/1.1 204 No Content
  Date: Wed, 25 Nov 2009 12:00:00 GMT
  Connection: close
  Server: AmazonS3
```

Related Resources

- GET Bucket tagging (p. 111)
- PUT Bucket tagging (p. 182)

DELETE Bucket website

Description

This operation removes the website configuration for a bucket. Amazon S3 returns a `200 OK` response upon successfully deleting a website configuration on the specified bucket. You will get a `200 OK` response if the website configuration you are trying to delete does not exist on the bucket. Amazon S3 returns a `404` response if the bucket specified in the request does not exist.

This `DELETE` operation requires the `S3:DeleteBucketWebsite` permission. By default, only the bucket owner can delete the *`website`* configuration attached to a bucket. However, bucket owners can grant other users permission to delete the *`website`* configuration by writing a bucket policy granting them the `S3:DeleteBucketWebsite` permission.

For more information about hosting websites, go to Hosting Websites on Amazon S3 in the *Amazon Simple Storage Service Developer Guide* .

Requests

Syntax

```
DELETE /?website HTTP/1.1
Host: bucketname.s3.amazonaws.com
Date: date
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))
```

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

This implementation of the operation uses only request headers that are common to all operations. For more information, see Common Request Headers (p. 12).

Request Elements

This operation does not use request elements.

Responses

Response Headers

This implementation of the operation uses only response headers that are common to most responses. For more information, see Common Response Headers (p. 14).

Response Elements

This implementation of the operation does not return response elements.

Examples

Sample Request

This request deletes the website configuration on the specified bucket.

```
DELETE ?website HTTP/1.1
Host: example-bucket.s3.amazonaws.com
Date: Thu, 27 Jan 2011 12:00:00 GMT
Authorization: signatureValue
```

Sample Response

```
HTTP/1.1 204 No Content
x-amz-id-2: aws-s3integ-s3ws-31008.sea31.amazon.com
x-amz-request-id: AF1DD829D3B49707
Date: Thu, 03 Feb 2011 22:10:26 GMT
Server: AmazonS3
```

Related Resources

- GET Bucket website (p. 131)
- PUT Bucket website (p. 191)

GET Bucket (List Objects)

Description

This implementation of the GET operation returns some or all (up to 1000) of the objects in a bucket. You can use the request parameters as selection criteria to return a subset of the objects in a bucket.

To use this implementation of the operation, you must have READ access to the bucket.

Note
To get a list of your buckets, see GET Service (p. 65).

Requests

Syntax

```
GET / HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))
```

Request Parameters

This implementation of GET uses the parameters in the following table to return a subset of the objects in a bucket.

Parameter	Description	Required
delimiter	A delimiter is a character you use to group keys. All keys that contain the same string between the *prefix*, if specified, and the first occurrence of the delimiter after the prefix are grouped under a single result element, *CommonPrefixes*. If you don't specify the *prefix* parameter, then the substring starts at the beginning of the key. The keys that are grouped under *CommonPrefixes* result element are not returned elsewhere in the response. Type: String Default: None	No
encoding-type	Requests Amazon S3 to encode the response and specifies the encoding method to use. An object key can contain any Unicode character; however, XML 1.0 parser cannot parse some characters, such as characters with an ASCII value from 0 to 10. For characters that are not supported in XML 1.0, you can add this parameter to request that Amazon S3 encode the keys in the response. Type: String Default: None Valid value: url	No

Parameter	Description	Required
marker	Specifies the key to start with when listing objects in a bucket. Amazon S3 returns object keys in alphabetical order, starting with key after the marker in order. Type: String Default: None	No
max-keys	Sets the maximum number of keys returned in the response body. You can add this to your request if you want to retrieve fewer than the default 1000 keys. The response might contain fewer keys but will never contain more. If there are additional keys that satisfy the search criteria but were not returned because *max-keys* was exceeded, the response contains `<IsTruncated>true</IsTruncated>`. To return the additional keys, see *marker*. Type: String Default: 1000	No
prefix	Limits the response to keys that begin with the specified prefix. You can use prefixes to separate a bucket into different groupings of keys. (You can think of using *prefix* to make groups in the same way you'd use a folder in a file system.) Type: String Default: None	No

Request Elements

This implementation of the operation does not use request elements.

Request Headers

This implementation of the operation uses only request headers that are common to all operations. For more information, see Common Request Headers (p. 12).

Responses

Response Headers

This implementation of the operation uses only response headers that are common to most responses. For more information, see Common Response Headers (p. 14).

Response Elements

Name	Description
`Contents`	Metadata about each object returned. Type: XML metadata Ancestor: ListBucketResult

Name	Description
`CommonPrefixes`	A response can contain *`CommonPrefixes`* only if you specify a *`delimiter`*. When you do, *`CommonPrefixes`* contains all (if there are any) keys between *`Prefix`* and the next occurrence of the string specified by *`delimiter`*. In effect, *`CommonPrefixes`* lists keys that act like subdirectories in the directory specified by *`Prefix`*. For example, if *`prefix`* is `notes/` and *`delimiter`* is a slash (/), in `notes/summer/july`, the common prefix is *`notes/summer/`*. All of the keys rolled up in a common prefix count as a single return when calculating the number of returns. See *`MaxKeys`*. Type: String Ancestor: ListBucketResult
`Delimiter`	Causes keys that contain the same string between the prefix and the first occurrence of the delimiter to be rolled up into a single result element in the *`CommonPrefixes`* collection. These rolled-up keys are not returned elsewhere in the response. Each rolled up result counts as only one return against the *`MaxKeys`* value. Type: String Ancestor: ListBucketResult
`DisplayName`	Object owner's name. Type: String Ancestor: ListBucketResult.Contents.Owner
`Encoding-Type`	Encoding type used by Amazon S3 to encode object key names in the XML response. If you specify `encoding-type` request parameter, Amazon S3 includes this element in the response, and returns encoded key name values in the following response elements: `Delimiter`, `Marker`, `Prefix`, `NextMarker`, `Key`. Type: String Ancestor: ListBucketResult
`ETag`	The entity tag is an MD5 hash of the object. The ETag only reflects changes to the contents of an object, not its metadata. Type: String Ancestor: ListBucketResult.Contents
`ID`	Object owner's ID. Type: String Ancestor: ListBucketResult.Contents.Owner
`IsTruncated`	Specifies whether (`true`) or not (`false`) all of the results were returned. All of the results may not be returned if the number of results exceeds that specified by *`MaxKeys`*. Type: Boolean Ancestor: ListBucketResult
`Key`	The object's key. Type: String Ancestor: ListBucketResult.Contents

Name	Description
`LastModified`	Date and time the object was last modified. Type: Date Ancestor: ListBucketResult.Contents
`Marker`	Indicates where in the bucket listing begins. `Marker` is included in the response if it was sent with the request. Type: String Ancestor: ListBucketResult
`MaxKeys`	The maximum number of keys returned in the response body. Type: String Ancestor: ListBucketResult
`Name`	Name of the bucket. Type: String Ancestor: ListBucketResult
`NextMarker`	When response is truncated (the `IsTruncated` element value in the response is true), you can use the key name in this field as `marker` in the subsequent request to get next set of objects. Amazon S3 lists objects in alphabetical order. **Note** This element is returned only if you have `delimiter` request parameter specified. If response does not include the `NextMaker` and it is truncated, you can use the value of the last `Key` in the response as the `marker` in the subsequent request to get the next set of object keys. Type: String Ancestor: ListBucketResult
`Owner`	Bucket owner. Type: String Children: DisplayName, ID Ancestor: ListBucketResult.Contents | CommonPrefixes
`Prefix`	Keys that begin with the indicated prefix. Type: String Ancestor: ListBucketResult
`Size`	Size in bytes of the object. Type: String Ancestor: ListBucketResult.Contents
`StorageClass`	STANDARD | REDUCED_REDUNDANCY | GLACIER Type: String Ancestor: ListBucketResult.Contents

Special Errors

This implementation of the operation does not return special errors. For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

Sample Request

This requests returns the objects in *BucketName*.

```
GET / HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: Wed, 12 Oct 2009 17:50:00 GMT
Authorization: authorization string
Content-Type: text/plain
```

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Name>bucket</Name>
    <Prefix/>
    <Marker/>
    <MaxKeys>1000</MaxKeys>
    <IsTruncated>false</IsTruncated>
    <Contents>
        <Key>my-image.jpg</Key>
        <LastModified>2009-10-12T17:50:30.000Z</LastModified>
        <ETag>&quot;fba9dede5f27731c9771645a39863328&quot;</ETag>
        <Size>434234</Size>
        <StorageClass>STANDARD</StorageClass>
        <Owner>
            <ID>75aa57f09aa0c8cae
ab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
            <DisplayName>mtd@amazon.com</DisplayName>
        </Owner>
    </Contents>
    <Contents>
       <Key>my-third-image.jpg</Key>
         <LastModified>2009-10-12T17:50:30.000Z</LastModified>
        <ETag>&quot;1b2cf535f27731c974343645a3985328&quot;</ETag>
        <Size>64994</Size>
        <StorageClass>STANDARD</StorageClass>
        <Owner>
            <ID>75aa57f09aa0c8cae
ab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
            <DisplayName>mtd@amazon.com</DisplayName>
        </Owner>
    </Contents>
</ListBucketResult>
```

Sample Request Using Request Parameters

This example lists up to 40 keys in the "quotes" bucket that start with "N" and occur lexicographically after "Ned".

```
GET /?prefix=N&marker=Ned&max-keys=40 HTTP/1.1
Host: quotes.s3.amazonaws.com
Date: Wed, 01 Mar  2006 12:00:00 GMT
Authorization: authorization string
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-id-2: gyB+3jRPnrkN98ZajxHXr3u7EFM67bNgSAxexeEHndCX/7GRnfTXxReKUQF28IfP
x-amz-request-id: 3B3C7C725673C630
Date: Wed, 01 Mar  2006 12:00:00 GMT
Content-Type: application/xml
Content-Length: 302
Connection: close
Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>quotes</Name>
  <Prefix>N</Prefix>
  <Marker>Ned</Marker>
  <MaxKeys>40</MaxKeys>
  <IsTruncated>false</IsTruncated>
  <Contents>
    <Key>Nelson</Key>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>&quot;828ef3fdfa96f00ad9f27c383fc9ac7f&quot;</ETag>
    <Size>5</Size>
    <StorageClass>STANDARD</StorageClass>
    <Owner>
      <ID>bcaf161ca5fb16fd081034f</ID>
      <DisplayName>webfile</DisplayName>
     </Owner>
  </Contents>
  <Contents>
    <Key>Neo</Key>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>&quot;828ef3fdfa96f00ad9f27c383fc9ac7f&quot;</ETag>
    <Size>4</Size>
    <StorageClass>STANDARD</StorageClass>
     <Owner>
      <ID>bcaf1ffd86a5fb16fd081034f</ID>
      <DisplayName>webfile</DisplayName>
    </Owner>
 </Contents>
</ListBucketResult>
```

Sample Request Using Prefix and Delimiter

Assume you have the following keys in your bucket.

```
sample.jpg

photos/2006/January/sample.jpg

photos/2006/February/sample2.jpg
```

```
photos/2006/February/sample3.jpg

photos/2006/February/sample4.jpg
```

The following GET request specifies the `delimiter` parameter with value `"/"`.

```
GET /?delimiter=/ HTTP/1.1
Host: example-bucket.s3.amazonaws.com
Date: Wed, 01 Mar  2006 12:00:00 GMT
Authorization: authorization string
```

The key *`sample.jpg`* does not contain the delimiter character, and Amazon S3 returns it in the *`Contents`* element in the response. However, all other keys contain the delimiter character. Amazon S3 groups these keys and return a single *`CommonPrefixes`* element with prefix value *`photos/`* that is a substring from the beginning of these keys to the first occurrence of the specified delimiter.

```
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>example-bucket</Name>
  <Prefix></Prefix>
  <Marker></Marker>
  <MaxKeys>1000</MaxKeys>
  <Delimiter>/</Delimiter>
  <IsTruncated>false</IsTruncated>
  <Contents>
    <Key>sample.html</Key>
    <LastModified>2011-02-26T01:56:20.000Z</LastModified>
    <ETag>&quot;bf1d737a4d46a19f3bced6905cc8b902&quot;</ETag>
    <Size>142863</Size>
    <Owner>
      <ID>canonical-user-id</ID>
      <DisplayName>display-name</DisplayName>
    </Owner>
    <StorageClass>STANDARD</StorageClass>
  </Contents>
  <CommonPrefixes>
    <Prefix>photos/</Prefix>
  </CommonPrefixes>
</ListBucketResult>
```

The following GET request specifies the `delimiter` parameter with value `"/"`, and the `prefix` parameter with value `photos/2006/`.

```
GET /?prefix=photos/2006/&delimiter=/ HTTP/1.1
Host: example-bucket.s3.amazonaws.com
Date: Wed, 01 Mar  2006 12:00:00 GMT
Authorization: authorization string
```

In response, Amazon S3 returns only the keys that start with the specified prefix. Further, it uses the *`delimiter`* character to group keys that contain the same substring until the first occurrence of the *`delimiter`* character after the specified prefix. For each such key group Amazon S3 returns one <CommonPrefixes> element in the response. The keys grouped under this *`CommonPrefixes`* element are not returned elsewhere in the response. The value returned in the `CommonPrefixes` element is a substring from the beginning of the key to the first occurrence of the specified delimiter after the prefix.

```
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>example-bucket</Name>
  <Prefix>photos/2006/</Prefix>
  <Marker></Marker>
  <MaxKeys>1000</MaxKeys>
  <Delimiter>/</Delimiter>
  <IsTruncated>false</IsTruncated>

  <CommonPrefixes>
    <Prefix>photos/2006/feb/</Prefix>
  </CommonPrefixes>
  <CommonPrefixes>
    <Prefix>photos/2006/jan/</Prefix>
  </CommonPrefixes>
</ListBucketResult>
```

Related Resources

- GET Object (p. 212)
- PUT Object (p. 250)
- PUT Bucket (p. 144)

GET Bucket acl

Description

This implementation of the GET operation uses the *acl* subresource to return the access control list (ACL) of a bucket. To use GET to return the ACL of the bucket, you must have READ_ACP access to the bucket. If READ_ACP permission is granted to the anonymous user, you can return the ACL of the bucket without using an authorization header.

Requests

Syntax

```
GET /?acl HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))
```

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

This implementation of the operation uses only request headers that are common to all operations. For more information, see Common Request Headers (p. 12).

Request Elements

This implementation of the operation does not use request elements.

Responses

Response Headers

This implementation of the operation uses only response headers that are common to most responses. For more information, see Common Response Headers (p. 14).

Response Elements

Name	Description
AccessControlList	Container for ACL information. Type: Container Ancestry: AccessControlPolicy
AccessControlPolicy	Container for the response. Type: Container Ancestry: None

Name	Description
`DisplayName`	Bucket owner's display name. This is returned only if the owner's e-mail address (or the forum name, if configured) can be determined from the *`ID`*. Type: String Ancestry: AccessControlPolicy.Owner
`Grant`	Container for *`Grantee`* and *`Permission`*. Type: Container Ancestry: AccessControlPolicy.AccessControlList
`Grantee`	Container for *`DisplayName`* and *`ID`* of the person being granted permissions. Type: Container Ancestry: AccessControlPolicy.AccessControlList.Grant
`ID`	Bucket owner's ID. Type: String Ancestry: AccessControlPolicy.Owner
`Owner`	Container for bucket owner information. Type: Container Ancestry: AccessControlPolicy
`Permission`	Permission given to the *`Grantee`* for bucket. Type: String Valid Values: FULL_CONTROL | WRITE | WRITE_ACP | READ | READ_ACP Ancestry: AccessControlPolicy.AccessControlList.Grant

Special Errors

This implementation of the operation does not return special errors. For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

Sample Request

The following request returns the ACL of the specified bucket.

```
GET ?acl HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: authorization string
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-id-2: eftixk72aD6Ap51TnqcoF8eFidJG9Z/2mkiDFu8yU9AS1ed4OpIszj7UDNEHGran
```

```
x-amz-request-id: 318BC8BC148832E5
Date: Wed, 28 Oct 2009 22:32:00 GMT
Last-Modified: Sun, 1 Jan 2006 12:00:00 GMT
Content-Length: 124
Content-Type: text/plain
Connection: close
Server: AmazonS3

<AccessControlPolicy>
  <Owner>
    <ID>75aa57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
    <DisplayName>CustomersName@amazon.com</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:type="CanonicalUser">
       <ID>75aa57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>

        <DisplayName>CustomersName@amazon.com</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

Related Resources

- GET Bucket Objects (p. 81)

GET Bucket cors

Description

Returns the *cors* configuration information set for the bucket.

To use this operation, you must have permission to perform the `s3:GetBucketCORS` action. By default, the bucket owner has this permission and can grant it to others.

To learn more `cors`, go to Enabling Cross-Origin Resource Sharing in the *Amazon Simple Storage Service Developer Guide*.

Requests

Syntax

```
GET /?cors HTTP/1.1
Host: bucketname.s3.amazonaws.com
Date: date
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))
```

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

This implementation of the operation uses only request headers that are common to all operations. For more information, see Common Request Headers (p. 12).

Request Elements

This implementation of the operation does not use request elements.

Responses

Response Headers

This implementation of the operation uses only response headers that are common to most responses. For more information, see Common Response Headers (p. 14).

Response Elements

This implementation of `GET` returns the following response elements.

Name	Description
CORSConfiguration	Container for up to 100 `CORSRules` elements. Type: Container Children: `CORSRules` Ancestor: None

Name	Description
CORSRule	A set of origins and methods (cross-origin access that you want to allow).. You can add up to 100 rules to the configuration. Type: Container Children: `AllowedOrigin`, `AllowedMethod`, `MaxAgeSeconds`, `ExposeHeader`, `ID`. Ancestor: `CORSConfiguration`
AllowedHeader	Specifies which headers are allowed in a pre-flight `OPTIONS` request through the `Access-Control-Request-Headers` header. Each header name specified in the `Access-Control-Request-Headers` must have a corresponding entry in the rule. Only the headers that were requested will be sent back. This element can contain at most one * wildcard character. A `CORSRule` can have at most one *MaxAgeSeconds* element. Type: Integer (seconds) Ancestor: `CORSRule`
AllowedMethod	Identifies an HTTP method that the domain/origin specified in the rule is allowed to execute. Each *CORSRule* must contain at least one *AllowedMethod* and one *AllowedOrigin* element. Type: Enum (GET, PUT, HEAD, POST, DELETE) Ancestor: `CORSRule`
AllowedOrigin	One or more response headers that you want customers to be able to access from their applications (for example, from a JavaScript XMLHttpRequest object). Each *CORSRule* must have at least one *AllowedOrigin* element. The string value can include at most one '*' wildcard character, for example, http://*.example.com". You can also specify only "*" to allow cross-origin access for all domains/origins. Type: String Ancestor: `CORSRule`
ExposeHeader	One or more headers in the response that you want customers to be able to access from their applications (for example, from a JavaScript `XMLHttpRequest` object). You add one *ExposeHeader* in the rule for each header. Type: String Ancestor: `CORSRule`
ID	An optional unique identifier for the rule. The ID value can be up to 255 characters long. The IDs help you find a rule in the configuration. Type: String Ancestor: `CORSRule`
MaxAgeSeconds	The time in seconds that your browser is to cache the preflight response for the specified resource. A `CORSRule` can have at most one *MaxAgeSeconds* element. Type: Integer (seconds) Ancestor: `CORSRule`

Special Errors

This implementation of the operation does not return special errors. For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

Example 1: Retrieve cors subresource

The following example gets the `cors` subresource of a bucket.

Sample Request

```
GET /?cors HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Date: Tue, 13 Dec 2011 19:14:42 GMT
Authorization: signatureValue
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-id-2: 0FmFIWsh/PpBuzZ0JFRC55ZGVmQW4SHJ7xVDqKwhEdJmf3q63RtrvH8ZuxW1Bol5
x-amz-request-id: 0CF038E9BCF63097
Date: Tue, 13 Dec 2011 19:14:42 GMT
Server: AmazonS3
Content-Length: 280

<CORSConfiguration>
     <CORSRule>
       <AllowedOrigin>http://www.example.com</AllowedOrigin>
       <AllowedMethod>GET</AllowedMethod>
       <MaxAgeSeconds>3000</MaxAgeSec>
       <ExposeHeader>x-amz-server-side-encryption</ExposeHeader>
     </CORSRule>
</CORSConfiguration>
```

Related Resources

- PUT Bucket cors (p. 157)
- DELETE Bucket cors (p. 71)
- OPTIONS object (p. 235)

GET Bucket lifecycle

Description

Returns the *lifecycle* configuration information set on the bucket. For information about lifecycle configuration, go to Object Lifecycle Management in the *Amazon Simple Storage Service Developer Guide*.

To use this operation, you must have permission to perform the `s3:GetLifecycleConfiguration` action. The bucket owner has this permission, by default. The bucket owner can grant this permission to others.

Requests

Syntax

```
GET /?lifecycle HTTP/1.1
Host: bucketname.s3.amazonaws.com
Date: date
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))
```

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

This implementation of the operation uses only request headers that are common to all operations. For more information, see Common Request Headers (p. 12).

Request Elements

This implementation of the operation does not use request elements.

Responses

Response Headers

This implementation of the operation uses only response headers that are common to most responses. For more information, see Common Response Headers (p. 14).

Response Elements

This implementation of `GET` returns the following response elements.

Name	Description	Required
Date	Specifies the date after which you want the corresponding action to take effect. When the action is in effect, Amazon S3 will perform the specific action on the applicable objects as they appear in the bucket (you identify applicable objects in the lifecycle `Rule` in which the action is defined). For example, suppose you add a `Transition` action to take effect on Dec. 31, 2014. Suppose this action applies to objects with the key prefix "documents/". When the action takes effect on this date, Amazon S3 transitions existing applicable objects to the GLACIER storage class. As long as the action is in effect, Amazon S3 will transition all objects that satisfy the prefix condition. The date value must conform to the ISO 8601 format. The time is always midnight UTC. Type: String Ancestor: `Expiration` or `Transition`	Yes, if `Days` is absent.
Days	Specifies the number of days after object creation when the specific rule action takes effect. The object's eligibility time is calculated as creation time + the number of days, and rounding the resulting time to the next day midnight UTC. Type: Non-negative Integer when used with `Transition`, Positive Integer when used with `Expiration`. Ancestor: `Transition` or `Expiration`.	Yes, if `Date` is absent.

Name	Description	Required
Expiration	This action specifies a period in the object's lifetime when Amazon S3 should take the appropriate expiration action. The expiration action occurs only on objects that are eligible according to the period specified in the child `Date` or `Days` element. The action Amazon S3 takes depends on whether the bucket is versioning enabled. • If versioning has never been enabled on the bucket, Amazon S3 deletes the only copy of the object permanently. • Otherwise, if your bucket is versioning-enabled (or versioning is suspended), the action applies only to the current version of the object. Buckets with versioning-enabled or versioning-suspended can have many versions of the same object, one current version, and zero or more noncurrent versions. Instead of deleting the current version, Amazon S3 makes it a noncurrent version by adding a delete marker as the new current version. **Important** If your bucket state is versioning-suspended, Amazon S3 creates a delete marker with version ID `null`. If you have a version with version ID `null`, then Amazon S3 overwrites that version. **Note** To set expiration for noncurrent objects, you must use the *NoncurrentVersionExpiration* action. Type: Container Children: Days or Date Ancestor: Rule	Yes, if no other action is present in the `Rule`.
ID	Unique identifier for the rule. The value cannot be longer than 255 characters. Type: String Ancestor: Rule	No
LifecycleConfiguration	Container for lifecycle rules. You can add as many as 1000 rules. Type: Container Children: Rule Ancestor: None	Yes

Name	Description	Required
NoncurrentDays	Specifies the number of days an object is noncurrent before Amazon S3 can perform the associated action. For information about the noncurrent days calculations, see Lifecycle Rules Based on the Number of Days in the *Amazon Simple Storage Service Developer Guide*. Type: Nonnegative Integer when used with `NoncurrentVersionTransition`, Positive Integer when used with `NoncurrentVersionExpiration`. Ancestor: `NoncurrentVersionExpiration` or `NoncurrentVersionTransition`	Yes, only if the ancestor is present.
NoncurrentVersionExpiration	Specifies when noncurrent object versions expire. Upon expiration, Amazon S3 permanently deletes the noncurrent object versions. You set this lifecycle configuration action on a bucket that has versioning enabled (or suspended) to request that Amazon S3 delete noncurrent object versions at a specific period in the object's lifetime. Type: Container Children: NoncurrentDays Ancestor: Rule	Yes, if no other action is present in the `Rule`.
NoncurrentVersionTransition	Container for the transition rule that describes when noncurrent objects transition to the GLACIER storage class. If your bucket is versioning-enabled (or versioning is suspended), you can set this action to request Amazon S3 to transition noncurrent object versions to the GLACIER storage class at a specific period in the object's lifetime. Type: Container Children: NoncurrentDays and StorageClass Ancestor: Rule	Yes, if no other action is present in the `Rule`.
Prefix	Object key prefix identifying one or more objects to which the rule applies. Type: String Ancestor: Rule	Yes
Rule	Container for a lifecycle rule. Type: Container Ancestor: LifecycleConfiguration	Yes
Status	If Enabled, Amazon S3 executes the rule as scheduled. If Disabled, Amazon S3 ignores the rule. Type: String Ancestor: Rule Valid values: Enabled or Disabled.	Yes

Name	Description	Required
StorageClass	Specifies the Amazon S3 storage class to which you want the object to transition to. Type: String Ancestor: Transition and NoncurrentVersionTransition Valid values: GLACIER.	Yes
Transition	This action specifies a period in the objects' lifetime when Amazon S3 should transition them to the GLACIER storage class. When this action is in effect, what Amazon S3 does depends on whether the bucket is versioning-enabled. • If versioning has never been enabled on the bucket, Amazon S3 transitions the only copy of the object to the GLACIER storage class. • Otherwise, when your bucket is versioning-enabled (or versioning is suspended), Amazon S3 transitions only the current versions of objects identified in the rule. **Note** A versioning-enabled or versioning-suspended bucket can have many versions of an object. This action has no impact on the noncurrent object versions. To transition noncurrent objects to the GLACIER storage class, you must use the *NoncurrentVersionTransition* action. Type: Container Children: Days or Date, and StorageClass Ancestor: Rule	Yes, if no other action is present in the `Rule`.

Special Errors

Error Code	Description	HTTP Status Code	SOAP Fault Code Prefix
`NoSuchLifecycleConfiguration`	The lifecycle configuration does not exist.	404 Not Found	Client

For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

Example 1: Retrieve lifecycle subresource

This example shows a GET request to retrieve the `lifecycle` subresource from the specified bucket and an example response with the returned lifecycle configuration.

Sample Request

```
GET /?lifecycle HTTP/1.1
Host: examplebucket.s3.amazonaws.com
x-amz-date: Thu, 15 Nov 2012 00:17:21 GMT
Authorization: signatureValue
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-id-2: ITnGT1y4RyTmXa3rPi4hklTXouTf0hccUjo0iCPjz6FnfIutBj3M7fPGlWO2SEWp
x-amz-request-id: 51991C342C575321
Date: Thu, 15 Nov 2012 00:17:23 GMT
Server: AmazonS3
Content-Length: 358

<?xml version="1.0" encoding="UTF-8"?>
<LifecycleConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Rule>
        <ID>Archive and then delete rule</ID>
        <Prefix>projectdocs/</Prefix>
        <Status>Enabled</Status>
        <Transition>
           <Days>365</Days>
           <StorageClass>GLACIER</StorageClass>
        </Transition>
        <Expiration>
           <Days>3650</Days>
        </Expiration>
    </Rule>
</LifecycleConfiguration>
```

Related Resources

- PUT Bucket lifecycle (p. 162)
- DELETE Bucket lifecycle (p. 73)

GET Bucket policy

Description

This implementation of the GET operation uses the *policy* subresource to return the policy of a specified bucket. To use this operation, you must have GetPolicy permissions on the specified bucket, and you must be the bucket owner.

If you don't have GetPolicy permissions, Amazon S3 returns a 403 Access Denied error. If you have the correct permissions, but you're not the bucket owner, Amazon S3 returns a 405 Method Not Allowed error. If the bucket does not have a policy, Amazon S3 returns a 404 Policy Not found error. There are restrictions about who can create bucket policies and which objects in a bucket they can apply to. For more information, go to Using Bucket Policies.

Requests

Syntax

```
GET /?policy HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))
```

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

This implementation of the operation uses only request headers that are common to all operations. For more information, see Common Request Headers (p. 12).

Request Elements

This implementation of the operation does not use request elements.

Responses

Response Headers

This implementation of the operation uses only response headers that are common to most responses. For more information, see Common Response Headers (p. 14).

Response Elements

The response contains the (JSON) policy of the specified bucket.

Special Errors

This implementation of the operation does not return special errors. For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

Sample Request

The following request returns the policy of the specified bucket.

```
GET ?policy HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: authorization string
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-id-2: Uuag1LuByru9pO4SAMPLEAtRPfTaOFg==
x-amz-request-id: 656c76696e67SAMPLE57374
Date: Tue, 04 Apr 2010 20:34:56 GMT
Connection: keep-alive
Server: AmazonS3


{
"Version":"2008-10-17",
"Id":"aaaa-bbbb-cccc-dddd",
"Statement" : [
    {
        "Effect":"Deny",
        "Sid":"1",
        "Principal" : {
            "AWS":["111122223333","444455556666"]
        },
        "Action":["s3:*"],
        "Resource":"arn:aws:s3:::bucket/*"
    }
 ]
}
```

Related Resources

- GET Bucket Objects (p. 81)

GET Bucket location

Description

This implementation of the GET operation uses the *location* subresource to return a bucket's Region. You set the bucket's Region using the *LocationConstraint* request parameter in a PUT *Bucket* request. For more information, see PUT Bucket (p. 144).

To use this implementation of the operation, you must be the bucket owner.

Requests

Syntax

```
GET /?location HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))
```

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

This implementation of the operation uses only request headers that are common to all operations. For more information, see Common Request Headers (p. 12).

Request Elements

This implementation of the operation does not use request elements.

Responses

Response Headers

This implementation of the operation uses only response headers that are common to most responses. For more information, see Common Response Headers (p. 14).

Response Elements

Name	Description
LocationConstraint	Specifies the Region where the bucket resides. For more information about region endpoints and location constraints, go to Regions and Endpoints in the *Amazon Web Services Glossary*. Type: String Valid Values: EU | eu-west-1 | us-west-1 | us-west-2 | ap-southeast-1 | ap-southeast-2 | ap-northeast-1 | sa-east-1 | empty string (for the US Classic Region) Ancestry: None

When the bucket's Region is US Classic, Amazon S3 returns an empty string for the bucket's Region:

```
<LocationConstraint xmlns="http://s3.amazonaws.com/doc/2006-03-01/"/>
```

Special Errors

This implementation of the operation does not return special errors. For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

Sample Request

The following request returns the Region of the specified bucket.

```
GET /?location HTTP/1.1
Host: myBucket.s3.amazonaws.com
Date: Tue, 09 Oct 2007 20:26:04 +0000
Authorization: signatureValue
```

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<LocationConstraint xmlns="http://s3.amazonaws.com/doc/2006-03-01/">EU</Loca
tionConstraint>
```

Related Resources

- GET Bucket Objects (p. 81)
- PUT Bucket (p. 144)

GET Bucket logging

Note
Logging functionality is currently in beta.

Description

This implementation of the `GET` operation uses the *`logging`* subresource to return the logging status of a bucket and the permissions users have to view and modify that status. To use `GET`, you must be the bucket owner.

Requests

Syntax

```
GET /?logging HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: authorization string
```

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

This implementation of the operation uses only request headers that are common to all operations. For more information, see Common Request Headers (p. 12).

Request Elements

This implementation of the operation does not use request elements.

Responses

Response Headers

This implementation of the operation uses only response headers that are common to most responses. For more information, see Common Response Headers (p. 14).

Response Elements

Name	Description
`BucketLoggingStatus`	Container for the response. Type: Container Ancestry: None
`EmailAddress`	E-mail address of the person whose logging permissions are displayed. Type: String Ancestry: BucketLoggingStatus.LoggingEnabled.TargetGrants.Grant.Grantee

Name	Description
`Grant`	Container for *`Grantee`* and *`Permission`*. Type: Container Ancestry: BucketLoggingStatus.LoggingEnabled.TargetGrants
`Grantee`	Container for *`EmailAddress`* of the person whose logging permissions are displayed. Type: Container Ancestry: BucketLoggingStatus.LoggingEnabled.TargetGrants.Grant
`LoggingEnabled`	Container for logging information. This element and its children are present when logging is enabled, otherwise, this element and its children are absent. Type: Container Ancestry: BucketLoggingStatus
`Permission`	Logging permissions assigned to the *`Grantee`* for the bucket. Type: String Valid Values: FULL_CONTROL | READ | WRITE Ancestry: BucketLoggingStatus.LoggingEnabled.TargetGrants.Grant
`TargetBucket`	Specifies the bucket whose logging status is being returned. This element specifies the bucket where server access logs will be delivered. Type: String Ancestry: BucketLoggingStatus.LoggingEnabled
`TargetGrants`	Container for granting information. Type: Container Ancestry: BucketLoggingStatus.LoggingEnabled
`TargetPrefix`	Specifies the prefix for the keys that the log files are being stored under. Type: String Ancestry: BucketLoggingStatus.LoggingEnabled

Special Errors

This implementation of the operation does not return special errors. For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

Sample Request

The following request returns the logging status for *`mybucket`*.

```
GET ?logging HTTP/1.1
Host: mybucket.s3.amazonaws.com
Date: Wed, 25 Nov 2009 12:00:00 GMT
Authorization: authorization string
```

Sample Response Showing an Enabled Logging Status

```
HTTP/1.1 200 OK
Date: Wed, 25 Nov 2009 12:00:00 GMT
Connection: close
Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<BucketLoggingStatus xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <LoggingEnabled>
    <TargetBucket>mybucketlogs</TargetBucket>
    <TargetPrefix>mybucket-access_log-/</TargetPrefix>
    <TargetGrants>
      <Grant>
        <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:type="AmazonCustomerByEmail">
          <EmailAddress>user@company.com</EmailAddress>
        </Grantee>
        <Permission>READ</Permission>
      </Grant>
    </TargetGrants>
  </LoggingEnabled>
</BucketLoggingStatus>
```

Sample Response Showing a Disabled Logging Status

```
HTTP/1.1 200 OK
Date: Wed, 25 Nov 2009 12:00:00 GMT
Connection: close
Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<BucketLoggingStatus xmlns="http://doc.s3.amazonaws.com/2006-03-01" />
```

Related Resources

- PUT Bucket (p. 144)
- PUT Bucket logging (p. 173)

GET Bucket notification

Description

This implementation of the GET operation uses the *notification* subresource to return the notification configuration of a bucket. Currently, the *s3:ReducedRedundancyLostObject* event is the only event supported for notifications. The *s3:ReducedRedundancyLostObject* event is triggered when Amazon S3 detects that it has lost all replicas of a Reduced Redundancy Storage object and can no longer service requests for that object.

If notifications are not enabled on the bucket, the operation returns an empty NotificatonConfiguration element.

By default, you must be the bucket owner to read the notification configuration of a bucket. However, the bucket owner can use a bucket policy to grant permission to other users to read this configuration with the *s3:GetBucketNotification* permission.

For more information about setting and reading the notification configuration on a bucket, see Setting Up Notification of Bucket Events. For more information about bucket policies, see Using Bucket Policies.

Requests

Syntax

```
GET /?notification HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))
```

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

This implementation of the operation uses only request headers that are common to all operations. For more information, see Common Request Headers (p. 12).

Request Elements

This implementation of the operation does not use request elements.

Responses

Response Headers

This implementation of the operation uses only response headers that are common to most responses. For more information, see Common Response Headers (p. 14).

Response Elements

Name	Description
NotificationConfiguration	Container for specifying the notification configuration of the bucket. If this element is empty, the bucket's notifications are turned off. Type: Container Children: *TopicConfiguration* Ancestry: None
TopicConfiguration	Container for specifying the topic configuration for the notification. Currently, only one topic can be configured for notifications. Type: Container Children: *Topic*, *Event* Ancestry: *NotificationConfiguration*
Topic	Amazon SNS topic to which Amazon S3 will publish a message to report the specified events for the bucket. Type: String Ancestry: *TopicConfiguration*
Event	Bucket event to send notifications for. Currently, *s3:ReducedRedundancyLostObject* is the only event supported for notifications. Type: String Valid Values: *s3:ReducedRedundancyLostObject* Ancestry: *TopicConfiguration*

Special Errors

This implementation of the operation does not return special errors. For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

Sample Request

This request returns the notification configuration on bucket quotes.s3.amazonaws.com.

```
GET ?notification HTTP/1.1
Host: quotes.s3.amazonaws.com
Date: Wed, 09 June 2010 12:00:00 GMT
Authorization: authorization string
```

Sample Response

This response returns that the notification configuration for the specified bucket.

```
HTTP/1.1 200 OK
x-amz-id-2: YgIPIfBiKa2bj0KMgUAdQkf3ShJTOOpXUueF6QKo
x-amz-request-id: 236A8905248E5A02
Date: Wed, 02 June 2010 12:00:00 GMT
Connection: close
Server: AmazonS3
<NotificationConfiguration>
   <TopicConfiguration>
       <Topic>arn:aws:sns:us-east-1:123456789012:myTopic</Topic>
       <Event>s3:ReducedRedundancyLostObject</Event>
   </TopicConfiguration>
</NotificationConfiguration>
```

Related Resources

- PUT Bucket notification (p. 178)

GET Bucket tagging

Description

This implementation of the GET operation uses the *tagging* subresource to return the tag set associated with the bucket.

To use this operation, you must have permission to perform the `s3:GetBucketTagging` action. By default, the bucket owner has this permission and can grant this permission to others.

Requests

Syntax

```
GET /?tagging HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))
```

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

This implementation of the operation uses only request headers that are common to all operations. For more information, see Common Request Headers (p. 12).

Request Elements

This implementation of the operation does not use request elements.

Responses

Response Headers

This implementation of the operation uses only response headers that are common to most responses. For more information, see Common Response Headers (p. 14).

Response Elements

Name	Description
`Tagging`	Contains the `TagSet` and `Tag` elements. Type: Container Ancestry: None
`TagSet`	Contains the tag set. Type: Container Ancestry: Tagging

Name	Description
`Tag`	Contains the tag information. Type: Container Ancestry: TagSet
`Key`	Name of the tag Type: String Ancestry: Tag
`Value`	Value of the tag Type: String Ancestry: Tag

Special Errors

- NoSuchTagSetError - There is no tag set associated with the bucket.

Examples

Sample Request

The following request returns the tag set of the specified bucket.

```
GET ?tagging HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: authorization string
```

Sample Response

```
HTTP/1.1 200 OK
Date: Wed, 25 Nov 2009 12:00:00 GMT
Connection: close
Server: AmazonS3

<Tagging>
  <TagSet>
     <Tag>
       <Key>Project</Key>
       <Value>Project One</Value>
     </Tag>
     <Tag>
       <Key>User</Key>
       <Value>jsmith</Value>
     </Tag>
  </TagSet>
</Tagging>
```

Related Resources

- PUT Bucket tagging (p. 182)

- DELETE Bucket tagging (p. 77)

GET Bucket Object versions

Description

You can use the *versions* subresource to list metadata about all of the versions of objects in a bucket. You can also use request parameters as selection criteria to return metadata about a subset of all the object versions. For more information, see Request Parameters (p. 114).

To use this operation, you must have READ access to the bucket.

Requests

Syntax

```
GET /?versions HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))
```

Request Parameters

This implementation of GET uses the parameters in the following table to return a subset of the objects in a bucket.

Parameter	Description	Required
delimiter	A delimiter is a character that you specify to group keys. All keys that contain the same string between the *prefix* and the first occurrence of the delimiter are grouped under a single result element in *CommonPrefixes*. These groups are counted as one result against the *max-keys* limitation. These keys are not returned elsewhere in the response. Also, see *prefix*. Type: String Default: None	No
encoding-type	Requests Amazon S3 to encode the response and specifies the encoding method to use. An object key can contain any Unicode character; however, XML 1.0 parser cannot parse some characters, such as characters with an ASCII value from 0 to 10. For characters that are not supported in XML 1.0, you can add this parameter to request that Amazon S3 encode the keys in the response. Type: String Default: None Valid value: url	No
key-marker	Specifies the key in the bucket that you want to start listing from. Also, see *version-id-marker*. Type: String Default: None	No

Parameter	Description	Required
max-keys	Sets the maximum number of keys returned in the response body. The response might contain fewer keys, but will never contain more. If additional keys satisfy the search criteria, but were not returned because *max-keys* was exceeded, the response contains <isTruncated>true</isTruncated>. To return the additional keys, see *key-marker* and *version-id-marker*. Type: String Default: 1000	No
prefix	Use this parameter to select only those keys that begin with the specified prefix. You can use prefixes to separate a bucket into different groupings of keys. (You can think of using *prefix* to make groups in the same way you'd use a folder in a file system.) You can use *prefix* with *delimiter* to roll up numerous objects into a single result under *CommonPrefixes*. Also, see *delimiter*. Type: String Default: None	No
version-id-marker	Specifies the object version you want to start listing from. Also, see *key-marker*. Type: String Default: None Valid Values: Valid version ID | Default Constraint: May not be an empty string	No

Request Headers

This implementation of the operation uses only request headers that are common to all operations. For more information, see Common Request Headers (p. 12).

Responses

Response Headers

This implementation of the operation uses only response headers that are common to most responses. For more information, see Common Response Headers (p. 14).

Response Elements

Name	Description
DeleteMarker	Container for an object that is a delete marker. Type: Container Children: Key, VersionId, IsLatest, LastModified, Owner Ancestor: ListVersionsResult
DisplayName	Object owner's name. Type: String Ancestor: ListVersionsResult.Version.Owner | ListVersionsResult.DeleteMarker.Owner

Name	Description
`Encoding-Type`	Encoding type used by Amazon S3 to encode object key names in the XML response. If you specify `encoding-type` request parameter, Amazon S3 includes this element in the response, and returns encoded key name values in the following response elements: `KeyMarker`, `NextKeyMarker`, `Prefix`, `Key`, and `Delimiter`. Type: String Ancestor: ListBucketResult
`ETag`	The entity tag is an MD5 hash of the object. The ETag only reflects changes to the contents of an object, not its metadata. Type: String Ancestor: ListVersionsResult.Version
`ID`	Object owner's ID. Type: String Ancestor: ListVersionsResult.Version.Owner | ListVersionsResult.DeleteMarker.Owner
`IsLatest`	Specifies whether the object is (`true`) or is not (`false`) the current version of an object. Type: Boolean Valid Values: `true` | `false` Ancestor: ListVersionsResult.Version | ListVersionsResult.DeleteMarker
`IsTruncated`	A flag that indicates whether (`true`) or not (`false`) Amazon S3 returned all of the results that satisfied the search criteria. If your results were truncated, you can make a follow-up paginated request using the *`NextKeyMarker`* and *`NextVersionIdMarker`* response parameters as a starting place in another request to return the rest of the results. Type: Boolean Valid Values: `true` | `false` Ancestor: ListVersionsResult
`Key`	The object's key. Type: String Ancestor: ListVersionsResult.Version | ListVersionsResult.DeleteMarker
`KeyMarker`	Marks the last *`Key`* returned in a truncated response. Type: String Ancestor: ListVersionsResult
`LastModified`	Date and time the object was last modified. Type: Date Ancestor: ListVersionsResult.Version | ListVersionsResult.DeleteMarker
`ListVersionsResult`	Container for the result. Type: Container Children: All elements in the response Ancestor: ListVersionsResult

Name	Description
`MaxKeys`	Specifies the maximum number of objects to return. Type: String Default: 1000 Valid Values: Integers from 1 to 1000, inclusive Ancestor: ListVersionsResult
`Name`	Bucket owner's name. Type: String Ancestor: ListVersionsResult
`NextKeyMarker`	When the number of responses exceeds the value of *`MaxKeys`*, *`NextKeyMarker`* specifies the first key not returned that satisfies the search criteria. Use this value for the *`key-marker`* request parameter in a subsequent request. Type: String Ancestor: ListVersionsResult
`NextVersionIdMarker`	When the number of responses exceeds the value of *`MaxKeys`*, *`NextVersionIdMarker`* specifies the first object version not returned that satisfies the search criteria. Use this value for the *`version-id-marker`* request parameter in a subsequent request. Type: String Ancestor: ListVersionsResult
`Owner`	Bucket owner. Type: String Children: DisplayName, ID Ancestor: ListVersionsResult.Version | ListVersionsResult.DeleteMarker
`Prefix`	Selects objects that start with the value supplied by this parameter. Type: String Ancestor: ListVersionsResult
`Size`	Size in bytes of the object. Type: String Ancestor: ListVersionsResult.Version
`StorageClass`	Always STANDARD. Type: String Ancestor: ListVersionsResult.Version
`Version`	Container for version information. Type: Container Ancestor: ListVersionsResult
`VersionId`	Version ID of an object Type: String Ancestor: ListVersionsResult.Version | ListVersionsResult.DeleteMarker

Name	Description
VersionIdMarker	Marks the last version of the *Key* returned in a truncated response. Type: String Ancestor: ListVersionsResult

Special Errors

This implementation of the operation does not return special errors. For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

Sample Request

The following request returns all of the versions of all of the objects in the specified bucket.

```
GET /?versions HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 +0000
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))
```

Sample Response to GET Versions

```
<?xml version="1.0" encoding="UTF-8"?>

<ListVersionsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01">
    <Name>bucket</Name>
    <Prefix>my</Prefix>
    <KeyMarker/>
    <VersionIdMarker/>
    <MaxKeys>5</MaxKeys>
    <IsTruncated>false</IsTruncated>
    <Version>
        <Key>my-image.jpg</Key>
        <VersionId>3/L4kqtJl40Nr8X8gdRQBpUMLUo</VersionId>
        <IsLatest>true</IsLatest>
         <LastModified>2009-10-12T17:50:30.000Z</LastModified>
        <ETag>&quot;fba9dede5f27731c9771645a39863328&quot;</ETag>
        <Size>434234</Size>
        <StorageClass>STANDARD</StorageClass>
        <Owner>
            <ID>75aa57f09aa0c8cae
ab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
            <DisplayName>mtd@amazon.com</DisplayName>
        </Owner>
    </Version>
    <DeleteMarker>
        <Key>my-second-image.jpg</Key>
        <VersionId>03jpff543dhffds434rfdsFDN943fdsFkdmqnh892</VersionId>
        <IsLatest>true</IsLatest>
        <LastModified>2009-11-12T17:50:30.000Z</LastModified>
        <Owner>
```

```
            <ID>75aa57f09aa0c8cae
ab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
            <DisplayName>mtd@amazon.com</DisplayName>
        </Owner>
    </DeleteMarker>
    <Version>
        <Key>my-second-image.jpg</Key>
        <VersionId>QUpfdndhfd8438MNFDN93jdnJFkdmqnh893</VersionId>
        <IsLatest>false</IsLatest>
        <LastModified>2009-10-10T17:50:30.000Z</LastModified>
        <ETag>&quot;9b2cf535f27731c974343645a3985328&quot;</ETag>
        <Size>166434</Size>
        <StorageClass>STANDARD</StorageClass>
        <Owner>
            <ID>75aa57f09aa0c8cae
ab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
            <DisplayName>mtd@amazon.com</DisplayName>
        </Owner>
    </Version>
    <DeleteMarker>
        <Key>my-third-image.jpg</Key>
        <VersionId>03jpff543dhffds434rfdsFDN943fdsFkdmqnh892</VersionId>
        <IsLatest>true</IsLatest>
        <LastModified>2009-10-15T17:50:30.000Z</LastModified>
        <Owner>
            <ID>75aa57f09aa0c8cae
ab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
            <DisplayName>mtd@amazon.com</DisplayName>
        </Owner>
    </DeleteMarker>
    <Version>
        <Key>my-third-image.jpg</Key>
        <VersionId>UIORUnfndfhnw89493jJFJ</VersionId>
        <IsLatest>false</IsLatest>
        <LastModified>2009-10-11T12:50:30.000Z</LastModified>
        <ETag>&quot;772cf535f27731c974343645a3985328&quot;</ETag>
        <Size>64</Size>
        <StorageClass>STANDARD</StorageClass>
        <Owner>
            <ID>75aa57f09aa0c8cae
ab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
            <DisplayName>mtd@amazon.com</DisplayName>
        </Owner>
     </Version>
</ListVersionsResult>
```

Sample Request

The following request returns objects in the order they were stored, returning the most recently stored object first starting with the value for *key-marker*.

```
GET /?versions&key-marker=key2 HTTP/1.1
Host: s3.amazonaws.com
Pragma: no-cache
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
Date: Thu, 10 Dec 2009 22:46:32 +0000
Authorization: signatureValue
```

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<ListVersionsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>mtp-versioning-fresh</Name>
  <Prefix/>
  <KeyMarker>key2</KeyMarker>
  <VersionIdMarker/>
  <MaxKeys>1000</MaxKeys>
  <IsTruncated>false</IsTruncated>
  <Version>
    <Key>key3</Key>
    <VersionId>I5VhmK6CDDdQ5Pwfe1gcHZWmHDpcv7gfmfc29UBxsKU.</VersionId>
    <IsLatest>true</IsLatest>
    <LastModified>2009-12-09T00:19:04.000Z</LastModified>
    <ETag>&quot;396fefef536d5ce46c7537ecf978a360&quot;</ETag>
    <Size>217</Size>
    <Owner>
     <ID>75aa57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>

    </Owner>
    <StorageClass>STANDARD</StorageClass>
  </Version>
  <DeleteMarker>
    <Key>sourcekey</Key>
    <VersionId>qDhprLU80sAlCFLu2DWgXAEDgKzWarn-HS_JU0TvYqs.</VersionId>
    <IsLatest>true</IsLatest>
    <LastModified>2009-12-10T16:38:11.000Z</LastModified>
    <Owner>
     <ID>75aa57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>

    </Owner>
  </DeleteMarker>
  <Version>
    <Key>sourcekey</Key>
    <VersionId>wxxQ7ezLaL5JN2Sislq66Syxxo0k7uHTUpb9qiiMxNg.</VersionId>
    <IsLatest>false</IsLatest>
    <LastModified>2009-12-10T16:37:44.000Z</LastModified>
    <ETag>&quot;396fefef536d5ce46c7537ecf978a360&quot;</ETag>
    <Size>217</Size>
    <Owner>
     <ID>75aa57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>

    </Owner>
    <StorageClass>STANDARD</StorageClass>
  </Version>
</ListVersionsResult>
```

Sample Request Using prefix

This example returns objects whose keys begin with `source`.

```
GET /?versions&prefix=source HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 +0000
Authorization: authorization string
```

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<ListVersionsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>mtp-versioning-fresh</Name>
  <Prefix>source</Prefix>
  <KeyMarker/>
  <VersionIdMarker/>
  <MaxKeys>1000</MaxKeys>
  <IsTruncated>false</IsTruncated>
  <DeleteMarker>
    <Key>sourcekey</Key>
    <VersionId>qDhprLU80sAlCFLu2DWgXAEDgKzWarn-HS_JU0TvYqs.</VersionId>
    <IsLatest>true</IsLatest>
    <LastModified>2009-12-10T16:38:11.000Z</LastModified>
    <Owner>
      <ID>75aa57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>

    </Owner>
  </DeleteMarker>
  <Version>
    <Key>sourcekey</Key>
    <VersionId>wxxQ7ezLaL5JN2Sislq66Syxxo0k7uHTUpb9qiiMxNg.</VersionId>
    <IsLatest>false</IsLatest>
    <LastModified>2009-12-10T16:37:44.000Z</LastModified>
    <ETag>&quot;396fefef536d5ce46c7537ecf978a360&quot;</ETag>
    <Size>217</Size>
    <Owner>
      <ID>75aa57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>

    </Owner>
    <StorageClass>STANDARD</StorageClass>
  </Version>
</ListVersionsResult>
```

Sample Request Using key-marker and version-id-marker Parameters

The following example returns objects starting at the specified key (*key-marker*) and version ID (*version-id-marker*).

```
GET /?versions&key-marker=key3&version-id-marker=t46ZenlYTZBnj HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 +0000
Authorization: signatureValue
```

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<ListVersionsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>mtp-versioning-fresh</Name>
  <Prefix/>
  <KeyMarker>key3</KeyMarker>
  <VersionIdMarker>t46ZenlYTZBnj</VersionIdMarker>
  <MaxKeys>1000</MaxKeys>
  <IsTruncated>false</IsTruncated>
```

```
  <DeleteMarker>
    <Key>sourcekey</Key>
    <VersionId>qDhprLU80sAlCFLu2DWgXAEDgKzWarn-HS_JU0TvYqs.</VersionId>
    <IsLatest>true</IsLatest>
    <LastModified>2009-12-10T16:38:11.000Z</LastModified>
    <Owner>
      <ID>75aa57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>

    </Owner>
  </DeleteMarker>
  <Version>
    <Key>sourcekey</Key>
    <VersionId>wxxQ7ezLaL5JN2Sislq66Syxxo0k7uHTUpb9qiiMxNg.</VersionId>
    <IsLatest>false</IsLatest>
    <LastModified>2009-12-10T16:37:44.000Z</LastModified>
    <ETag>&quot;396fefef536d5ce46c7537ecf978a360&quot;</ETag>
    <Size>217</Size>
    <Owner>
      <ID>75aa57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>

    </Owner>
    <StorageClass>STANDARD</StorageClass>
  </Version>
</ListVersionsResult>
```

Sample Request Using key-marker, version-id-marker and max-keys

The following request returns up to three (the value of *`max-keys`*) objects starting with the key specified by *`key-marker`* and the version ID specified by *`version-id-marker`*.

```
GET /?versions&key-marker=key3&version-id-marker=t46Z0menlYTZBnj HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 +0000
Authorization: authorization string
```

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<ListVersionsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>mtp-versioning-fresh</Name>
  <Prefix/>
  <KeyMarker>key3</KeyMarker>
  <VersionIdMarker>null</VersionIdMarker>
  <NextKeyMarker>key3</NextKeyMarker>
  <NextVersionIdMarker>d-d309mfjFrUmoQ0DBsVqmcMV15OI.</NextVersionIdMarker>
  <MaxKeys>2</MaxKeys>
  <IsTruncated>true</IsTruncated>
  <Version>
    <Key>key3</Key>
    <VersionId>8XECiENpj8pydEDJdd-_VRrvaGKAHOaGMNW7tg6UViI.</VersionId>
    <IsLatest>false</IsLatest>
    <LastModified>2009-12-09T00:18:23.000Z</LastModified>
    <ETag>&quot;396fefef536d5ce46c7537ecf978a360&quot;</ETag>
    <Size>217</Size>
    <Owner>
```

```
     <ID>75aa57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>

    </Owner>
    <StorageClass>STANDARD</StorageClass>
  </Version>
  <Version>
    <Key>key3</Key>
    <VersionId>d-d309mfjFri40QYukDozqBt3UmoQ0DBsVqmcMV15OI.</VersionId>
    <IsLatest>false</IsLatest>
    <LastModified>2009-12-09T00:18:08.000Z</LastModified>
    <ETag>&quot;396fefef536d5ce46c7537ecf978a360&quot;</ETag>
    <Size>217</Size>
    <Owner>
      <ID>75aa57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>

    </Owner>
    <StorageClass>STANDARD</StorageClass>
  </Version>
</ListVersionsResult>
```

Sample Request Using the Delimiter and the Prefix Parameters

Assume you have the following keys in your bucket, `example-bucket`.

`photos/2006/January/sample.jpg`

`photos/2006/February/sample.jpg`

`photos/2006/March/sample.jpg`

`videos/2006/March/sample.wmv`

`sample.jpg`

The following GET versions request specifies the delimiter parameter with value "/".

```
GET /?versions&delimiter=/ HTTP/1.1
Host: example-bucket.s3.amazonaws.com
Date: Wed, 02 Feb 2011 20:34:56 GMT
Authorization: authorization string
```

The list of keys from the specified bucket are shown in the following response.

The response returns the `sample.jpg` key in a <Version> element. However, because all the other keys contain the specified delimiter, a distinct substring, from the beginning of the key to the first occurrence of the delimiter, from each of these keys is returned in a <CommonPrefixes> element. The key substrings, `photos/` and `videos/`, in the <CommonPrefixes> element indicate that there are one or more keys with these key prefixes.

This is a useful scenario if you use key prefixes for your objects to create a logical folder like structure. In this case you can interpret the result as the folders `photos/` and `videos/` have one or more objects.

```
<ListVersionsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>mvbucketwithversionon1</Name>
  <Prefix></Prefix>
  <KeyMarker></KeyMarker>
```

```
  <VersionIdMarker></VersionIdMarker>
  <MaxKeys>1000</MaxKeys>
  <Delimiter>/</Delimiter>
  <IsTruncated>false</IsTruncated>

  <Version>
    <Key>Sample.jpg</Key>
    <VersionId>toxMzQlBsGyGCz1YuMWMp90cdXLzqOCH</VersionId>
    <IsLatest>true</IsLatest>
    <LastModified>2011-02-02T18:46:20.000Z</LastModified>
    <ETag>&quot;3305f2cfc46c0f04559748bb039d69ae&quot;</ETag>
    <Size>3191</Size>
    <Owner>
     <ID>852b113e7a2f25102679df27bb0ae12b3f85be6f290b936c4393484be31bebcc</ID>

      <DisplayName>display-name</DisplayName>
    </Owner>
    <StorageClass>STANDARD</StorageClass>
  </Version>

  <CommonPrefixes>
    <Prefix>photos/</Prefix>
  </CommonPrefixes>
  <CommonPrefixes>
    <Prefix>videos/</Prefix>
  </CommonPrefixes>
</ListVersionsResult>
```

In addition to the delimiter parameter you can filter results by adding a `prefix` parameter as shown in the following request.

```
GET /?versions&prefix=photos/2006/&delimiter=/ HTTP/1.1
Host: example-bucket.s3.amazonaws.com
Date: Wed, 02 Feb 2011 19:34:02 GMT
Authorization: authorization string
```

In this case the response will include only objects keys that start with the specified prefix. The value returned in the <CommonPrefixes> element is a substring from the beginning of the key to the first occurrence of the specified delimiter after the prefix.

```
<?xml version="1.0" encoding="UTF-8"?>
<ListVersionsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>example-bucket</Name>
  <Prefix>photos/2006/</Prefix>
  <KeyMarker></KeyMarker>
  <VersionIdMarker></VersionIdMarker>
  <MaxKeys>1000</MaxKeys>
  <Delimiter>/</Delimiter>
  <IsTruncated>false</IsTruncated>
  <Version>
    <Key>photos/2006/</Key>
    <VersionId>3U275dAA4gz8ZOqOPHtJCUOi60krpCdy</VersionId>
    <IsLatest>true</IsLatest>
    <LastModified>2011-02-02T18:47:27.000Z</LastModified>
    <ETag>&quot;d41d8cd98f00b204e9800998ecf8427e&quot;</ETag>
    <Size>0</Size>
```

```
    <Owner>
      <ID>75aa57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>

       <DisplayName>display-name</DisplayName>
    </Owner>
    <StorageClass>STANDARD</StorageClass>
  </Version>
  <CommonPrefixes>
    <Prefix>photos/2006/February/</Prefix>
  </CommonPrefixes>
  <CommonPrefixes>
    <Prefix>photos/2006/January/</Prefix>
  </CommonPrefixes>
  <CommonPrefixes>
    <Prefix>photos/2006/March/</Prefix>
  </CommonPrefixes>
</ListVersionsResult>
```

Related Resources

- GET Bucket Objects (p. 81)
- GET Object (p. 212)
- PUT Object (p. 250)
- DELETE Object (p. 200)

GET Bucket requestPayment

Description

This implementation of the GET operation uses the *requestPayment* subresource to return the request payment configuration of a bucket. To use this version of the operation, you must be the bucket owner. For more information, see Requester Pays Buckets.

Requests

Syntax

```
GET ?requestPayment HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: Date
Authorization: authorization string
```

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

This implementation of the operation uses only request headers that are common to all operations. For more information, see Common Request Headers (p. 12).

Responses

Response Headers

This implementation of the operation uses only response headers that are common to most responses. For more information, see Common Response Headers (p. 14).

Response Elements

Name	Description
Payer	Specifies who pays for the download and request fees. Type: Enum Valid Values: Requester | BucketOwner Ancestor: RequestPaymentConfiguration
RequestPaymentConfiguration	Container for *Payer*. Type: Container

Special Errors

This implementation of the operation does not return special errors. For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

Sample Request

The following request returns the payer for the bucket, `colorpictures`.

```
GET ?requestPayment HTTP/1.1
Host: colorpictures.s3.amazonaws.com
Date: Wed, 01 Mar 2009 12:00:00 GMT
Authorization: authorization string
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-id-2: YgIPIfBiKa2bj0KMg95r/0zo3emzU4dzsD4rcKCHQUAdQkf3ShJTOOpXUueF6QKo
x-amz-request-id: 236A8905248E5A01
Date: Wed, 01 Mar 2009 12:00:00 GMT
Content-Type: [type]
Content-Length: 0
Connection: close
Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<RequestPaymentConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Payer>Requester</Payer>
</RequestPaymentConfiguration>
```

This response shows that the bucket is a Requester Pays bucket, meaning the person requesting a download from this bucket pays the transfer fees.

Related Resources

- GET Bucket (List Objects) (p. 81)

GET Bucket versioning

Description

This implementation of the GET operation uses the *versioning* subresource to return the versioning state of a bucket. To retrieve the versioning state of a bucket, you must be the bucket owner.

This implementation also returns the MFA Delete status of the versioning state, i.e., if the MFA Delete status is `enabled`, the bucket owner must use an authentication device to change the versioning state of the bucket.

There are three versioning states:

- If you enabled versioning on a bucket, the response is:

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Status>Enabled</Status>
</VersioningConfiguration>
```

- If you suspended versioning on a bucket, the response is:

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Status>Suspended</Status>
</VersioningConfiguration>
```

- If you never enabled (or suspended) versioning on a bucket, the response is:

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/"/>
```

Requests

Syntax

```
GET /?versioning HTTP/1.1
Host: BucketName.s3.amazonaws.com
Content-Length: length
Date: date
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))
```

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

This implementation of the operation uses only request headers that are common to all operations. For more information, see Common Request Headers (p. 12).

Request Elements

This implementation of the operation does not use request elements.

Responses

Response Headers

This implementation of the operation uses only response headers that are common to most responses. For more information, see Common Response Headers (p. 14).

Response Elements

This implementation of `GET` returns the following response elements.

Name	Description
`MfaDelete`	Specifies whether MFA delete is enabled in the bucket versioning configuration. This element is only returned if the bucket has been configured with MfaDelete. If the bucket has never been so configured, this element is not returned. Type: Enum Valid Values: Disabled | Enabled Ancestor: VersioningConfiguration
`Status`	The versioning state of the bucket. Type: Enum Valid Values: Suspended | Enabled Ancestor: VersioningConfiguration
`VersioningConfiguration`	Container for the *`Status`* response element. Type: Container Ancestor: None

Special Errors

This implementation of the operation does not return special errors. For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

Sample Request

This example returns the versioning state of `myBucket`.

```
GET /?versioning HTTP/1.1
Host: myBucket.s3.amazonaws.com
Date: Wed, 12 Oct 2009 17:50:00 GMT
Authorization: authorization string
Content-Type: text/plain
```

Sample Response

The following is a sample of the response body (only) that shows bucket versioning is enabled.

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Status>Enabled</Status>
</VersioningConfiguration>
```

Related Resources

- GET Object (p. 212)
- PUT Object (p. 250)
- DELETE Object (p. 200)

GET Bucket website

Description

This implementation of the `GET` operation returns the website configuration associated with a bucket. To host website on Amazon S3, you can configure a bucket as website by adding a website configuration. For more information about hosting websites, go to Hosting Websites on Amazon S3 in the *Amazon Simple Storage Service Developer Guide* .

This `GET` operation requires the `S3:GetBucketWebsite` permission. By default, only the bucket owner can read the bucket *`website`* configuration. However, bucket owners can allow other users to read the *`website`* configuration by writing a bucket policy granting them the `S3:GetBucketWebsite` permission.

Requests

Syntax

```
GET /?website HTTP/1.1
Host: bucketname.s3.amazonaws.com
Date: date
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))
```

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

This implementation of the operation uses only request headers that are common to all operations. For more information, see Common Request Headers (p. 12).

Request Elements

This operation does not use request elements.

Responses

Response Headers

This implementation of the operation uses only response headers that are common to most responses. For more information, see Common Response Headers (p. 14).

Response Elements

The response XML includes same elements that were uploaded when you configured the bucket as website. For more information, see PUT Bucket website (p. 191).

Examples

Sample Request

This request retrieves website configuration on the specified bucket.

```
GET ?website HTTP/1.1
Host: example-bucket.s3.amazon.com
Date: Thu, 27 Jan 2011 00:49:20 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:n0Nhek72Ufg/u7Sm5C1dqRLs8XX=
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-id-2: YgIPIfBiKa2bj0KMgUAdQkf3ShJTOOpXUueF6QKo
x-amz-request-id: 3848CD259D811111
Date: Thu, 27 Jan 2011 00:49:26 GMT
Content-Length: 240
Content-Type: application/xml
Transfer-Encoding: chunked
Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<WebsiteConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <IndexDocument>
    <Suffix>index.html</Suffix>
  </IndexDocument>
  <ErrorDocument>
    <Key>404.html</Key>
  </ErrorDocument>
</WebsiteConfiguration>
```

Related Resources

- DELETE Bucket website (p. 79)
- PUT Bucket website (p. 191)

HEAD Bucket

Description

This operation is useful to determine if a bucket exists and you have permission to access it. The operation returns a `200 OK` if the bucket exists and you have permission to access it. Otherwise, the operation might return responses such as `404 Not Found` and `403 Forbidden`.

For information about permissions required for this bucket operation, go to Specifying Permissions in a Policy in the *Amazon Simple Storage Service Developer Guide*.

Requests

Syntax

```
HEAD / HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))
```

Request Parameters

This implementation of the operation does not use request parameters.

Request Elements

This implementation of the operation does not use request elements.

Request Headers

This implementation of the operation uses only request headers that are common to all operations. For more information, see Common Request Headers (p. 12).

Responses

Response Headers

This implementation of the operation uses only response headers that are common to most responses. For more information, see Common Response Headers (p. 14).

Response Elements

This implementation of the operation does not return response elements.

Special Errors

This implementation of the operation does not return special errors. For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

Sample Request

This requests returns the objects in *BucketName*.

```
HEAD / HTTP/1.1
Date: Fri, 10 Feb 2012 21:34:55 GMT
Authorization: authorization string
Host: myawsbucket.s3.amazonaws.com
Connection: Keep-Alive
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-id-2: JuKZqmXuiwFeDQxhD7M8KtsKobSzWA1QEjLbTMTagkKdBX2z7Il/jGhDeJ3j6s80
x-amz-request-id: 32FE2CEB32F5EE25
Date: Fri, 10 2012 21:34:56 GMT
Server: AmazonS3
```

List Multipart Uploads

Description

This operation lists in-progress multipart uploads. An in-progress multipart upload is a multipart upload that has been initiated, using the Initiate Multipart Upload request, but has not yet been completed or aborted.

This operation returns at most 1,000 multipart uploads in the response. 1,000 multipart uploads is the maximum number of uploads a response can include, which is also the default value. You can further limit the number of uploads in a response by specifying the *`max-uploads`* parameter in the response. If additional multipart uploads satisfy the list criteria, the response will contain an `IsTruncated` element with the value `true`. To list the additional multipart uploads, use the `key-marker` and `upload-id-marker` request parameters.

In the response, the uploads are sorted by key. If your application has initiated more than one multipart upload using the same object key, then uploads in the response are first sorted by key. Additionally, uploads are sorted in ascending order within each key by the upload initiation time.

For more information on multipart uploads, go to Uploading Objects Using Multipart Upload in the *Amazon S3 Developer Guide*.

For information on permissions required to use the multipart upload API, go to Multipart Upload API and Permissions in the *Amazon Simple Storage Service Developer Guide* .

Requests

Syntax

```
GET /?uploads HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: Date
Authorization: authorization string
```

Request Parameters

Parameter	Description	Required
`delimiter`	Character you use to group keys. All keys that contain the same string between the *`prefix`*, if specified, and the first occurrence of the delimiter after the prefix are grouped under a single result element, *`CommonPrefixes`*. If you don't specify the *`prefix`* parameter, then the substring starts at the beginning of the key. The keys that are grouped under *`CommonPrefixes`* result element are not returned elsewhere in the response. Type: String	No

Parameter	Description	Required
encoding-type	Requests Amazon S3 to encode the response and specifies the encoding method to use. An object key can contain any Unicode character; however, XML 1.0 parser cannot parse some characters, such as characters with an ASCII value from 0 to 10. For characters that are not supported in XML 1.0, you can add this parameter to request that Amazon S3 encode the keys in the response. Type: String Default: None Valid value: `url`	No
max-uploads	Sets the maximum number of multipart uploads, from 1 to 1,000, to return in the response body. 1,000 is the maximum number of uploads that can be returned in a response. Type: Integer Default: 1,000	No
key-marker	Together with `upload-id-marker`, this parameter specifies the multipart upload after which listing should begin. If `upload-id-marker` is not specified, only the keys lexicographically greater than the specified *key-marker* will be included in the list. If `upload-id-marker` is specified, any multipart uploads for a key equal to the *key-marker* might also be included, provided those multipart uploads have upload IDs lexicographically greater than the specified *upload-id-marker*. Type: String	No
prefix	Lists in-progress uploads only for those keys that begin with the specified prefix. You can use prefixes to separate a bucket into different grouping of keys. (You can think of using prefix to make groups in the same way you'd use a folder in a file system.) Type: String	No
upload-id-marker	Together with `key-marker`, specifies the multipart upload after which listing should begin. If `key-marker` is not specified, the `upload-id-marker` parameter is ignored. Otherwise, any multipart uploads for a key equal to the `key-marker` might be included in the list only if they have an upload ID lexicographically greater than the specified *upload-id-marker*. Type: String	No

Request Headers

This operation uses only Request Headers common to most requests. For more information, see Common Request Headers (p. 12).

Request Elements

This operation does not use request elements.

Responses

Response Headers

This operation uses only response headers that are common to most responses. For more information, see Common Response Headers (p. 14).

Response Elements

Name	Description
`ListMultipartUploadsResult`	Container for the response. Children: *`Bucket`*, *`KeyMarker`*, *`UploadIdMarker`*, *`NextKeyMarker`*, *`NextUploadIdMarker`*, *`MaxUploads`*, *`Delimiter`*, *`Prefix`*, *`CommonPrefixes`*, *`IsTruncated`* Type: Container Ancestor: None
`Bucket`	Name of the bucket to which the multipart upload was initiated. Type: `String` Ancestor: *`ListMultipartUploadsResult`*
`KeyMarker`	The key at or after which the listing began. Type: `String` Ancestor: *`ListMultipartUploadsResult`*
`UploadIdMarker`	Upload ID after which listing began. Type: `String` Ancestor: *`ListMultipartUploadsResult`*
`NextKeyMarker`	When a list is truncated, this element specifies the value that should be used for the *`key-marker`* request parameter in a subsequent request. Type: `String` Ancestor: *`ListMultipartUploadsResult`*
`NextUploadIdMarker`	When a list is truncated, this element specifies the value that should be used for the *`upload-id-marker`* request parameter in a subsequent request. Type: `String` Ancestor: *`ListMultipartUploadsResult`*
`Encoding-Type`	Encoding type used by Amazon S3 to encode object key names in the XML response. If you specify `encoding-type` request parameter, Amazon S3 includes this element in the response, and returns encoded key name values in the following response elements: `Delimiter`, `KeyMarker`, `Prefix`, `NextKeyMarker`, `Key`. Type: String Ancestor: ListBucketResult

Name	Description
`MaxUploads`	Maximum number of multipart uploads that could have been included in the response. Type: `Integer` Ancestor: *`ListMultipartUploadsResult`*
`IsTruncated`	Indicates whether the returned list of multipart uploads is truncated. A value of `true` indicates that the list was truncated. The list can be truncated if the number of multipart uploads exceeds the limit allowed or specified by `MaxUploads`. Type: `Boolean` Ancestor: `ListMultipartUploadsResult`
`Upload`	Container for elements related to a particular multipart upload. A response can contain zero or more *`Upload`* elements. Type: Container Children: *`Key`*, *`UploadId`*, *`InitiatorOwner`*, *`StorageClass`*, *`Initiated`* Ancestor: *`ListMultipartUploadsResult`*
`Key`	Key of the object for which the multipart upload was initiated. Type: *`Integer`* Ancestor: *`Upload`*
`UploadId`	Upload ID that identifies the multipart upload. Type: Integer Ancestor: *`Upload`*
`Initiator`	Container element that identifies who initiated the multipart upload. If the initiator is an AWS account, this element provides the same information as the *`Owner`* element. If the initiator is an IAM User, then this element provides the user ARN and display name. Children: *`ID`*, *`DisplayName`* Type: Container Ancestor: *`Upload`*
`ID`	If the principal is an AWS account, it provides the Canonical User ID. If the principal is an IAM User, it provides a user ARN value. Type: String Ancestor: *`Initiator`*, *`Owner`*
`DisplayName`	Principal's name. Type: String Ancestor: *`Initiator`* , *`Owner`*

Name	Description
`Owner`	Container element that identifies the object owner, after the object is created. If multipart upload is initiated by an IAM user, this element provides a the parent account ID and display name. Type: Container Children: *`ID`*, *`DisplayName`* Ancestor: *`Upload`*
`StorageClass`	The class of storage (`STANDARD` or `REDUCED_REDUDANCY`) that will be used to store the object when the multipart upload is complete. Type: String Ancestor: *`Upload`*
`Initiated`	Date and time at which the multipart upload was initiated. Type: *`Date`* Ancestor: *`Upload`*
`ListMultipartUploadsResult.Prefix`	When a prefix is provided in the request, this field contains the specified prefix. The result contains only keys starting with the specified prefix. Type: *`String`* Ancestor: *`ListMultipartUploadsResult`*
`Delimiter`	Contains the delimiter you specified in the request. If you don't specify a delimiter in your request, this element is absent from the response. Type: *`String`* Ancestor: *`ListMultipartUploadsResult`*
`CommonPrefixes`	If you specify a delimiter in the request, then the result returns each distinct key prefix containing the delimiter in a `CommonPrefixes` element. The distinct key prefixes are returned in the `Prefix` child element. Type: *`Container`* Ancestor: *`ListMultipartUploadsResult`*
`CommonPrefixes.Prefix`	If the request does not include the `Prefix` parameter, then this element shows only the substring of the key that precedes the first occurrence of the delimiter character. These keys are not returned anywhere else in the response. If the request includes the `Prefix` parameter, then this element shows the substring of the key from the beginning to the first occurrence of the delimiter after the prefix. Type: *`String`* Ancestor: *`CommonPrefixes`*

Examples

Sample Request

The following request lists three multipart uploads. The request specifies the `max-uploads` request parameter to set the maximum number of multipart uploads to return in the response body.

```
GET /?uploads&max-uploads=3 HTTP/1.1
Host: example-bucket.s3.amazonaws.com
Date: Mon, 1 Nov 2010 20:34:56 GMT
Authorization: authorization string
```

Sample Response

The following sample response indicates that the multipart upload list was truncated and provides the *NextKeyMarker* and the *NextUploadIdMarker* elements. You specify these values in your subsequent requests to read the next set of multipart uploads. That is, send a subsequent request specifying `key-marker=my-movie2.m2ts` (value of the *NextKeyMarker* element) and `upload-id-marker=YW55IG-lkZWEgd2h5IGVsdmluZydzIHVwbG9hZCBmYWlsZWQ` (value of the *NextUploadIdMarker*).

The sample response also shows a case of two multipart uploads in progress with the same key (*my-movie.m2ts*). That is, the response shows two uploads with the same key. This response shows the uploads sorted by key, and within each key the uploads are sorted in ascending order by the time the multipart upload was initiated.

```
HTTP/1.1 200 OK
x-amz-id-2: Uuag1LuByRx9e6j5Onimru9pO4ZVKnJ2Qz7/C1NPcfTWAtRPfTaOFg==
x-amz-request-id: 656c76696e6727732072657175657374
Date: Mon, 1 Nov 2010 20:34:56 GMT
Content-Length: 1330
Connection: keep-alive
Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<ListMultipartUploadsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Bucket>bucket</Bucket>
  <KeyMarker></KeyMarker>
  <UploadIdMarker></UploadIdMarker>
  <NextKeyMarker>my-movie.m2ts</NextKeyMarker>
  <NextUploadIdMarker>YW55IGlkZWEgd2h5IGVsdmluZydzIHVwbG9hZCBmYWlsZWQ</NextUp
loadIdMarker>
  <MaxUploads>3</MaxUploads>
  <IsTruncated>true</IsTruncated>
  <Upload>
    <Key>my-divisor</Key>
    <UploadId>XMgbGlrZSBlbHZpbmcncyBub3QgaGF2aW5nIG11Y2ggbHVjaw</UploadId>
    <Initiator>
      <ID>arn:aws:iam::111122223333:user/user1-11111a31-17b5-4fb7-9df5-
b111111f13de</ID>
      <DisplayName>user1-11111a31-17b5-4fb7-9df5-b111111f13de</DisplayName>
    </Initiator>
    <Owner>
     <ID>75aa57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>

      <DisplayName>OwnerDisplayName</DisplayName>
    </Owner>
```

```
    <StorageClass>STANDARD</StorageClass>
    <Initiated>2010-11-10T20:48:33.000Z</Initiated>
  </Upload>
  <Upload>
    <Key>my-movie.m2ts</Key>
    <UploadId>VXBsb2FkIElEIGZvciBlbHZpbmcncyBteS1tb3ZpZS5tMnRzIHVwbG9hZA</Up
loadId>
    <Initiator>
      <ID>b1d16700c70b0b05597d7acd6a3f92be</ID>
      <DisplayName>InitiatorDisplayName</DisplayName>
    </Initiator>
    <Owner>
      <ID>b1d16700c70b0b05597d7acd6a3f92be</ID>
      <DisplayName>OwnerDisplayName</DisplayName>
    </Owner>
    <StorageClass>STANDARD</StorageClass>
    <Initiated>2010-11-10T20:48:33.000Z</Initiated>
  </Upload>
  <Upload>
    <Key>my-movie.m2ts</Key>
    <UploadId>YW55IGlkZWEgd2h5IGVsdmluZydzIHVwbG9hZCBmYWlsZWQ</UploadId>
    <Initiator>
      <ID>arn:aws:iam::444455556666:user/user1-22222a31-17b5-4fb7-9df5-
b222222f13de</ID>
      <DisplayName>user1-22222a31-17b5-4fb7-9df5-b222222f13de</DisplayName>
    </Initiator>
    <Owner>
      <ID>b1d16700c70b0b05597d7acd6a3f92be</ID>
      <DisplayName>OwnerDisplayName</DisplayName>
    </Owner>
    <StorageClass>STANDARD</StorageClass>
    <Initiated>2010-11-10T20:49:33.000Z</Initiated>
  </Upload>
</ListMultipartUploadsResult>
```

Sample Request Using the Delimiter and the Prefix Parameters

Assume you have a multipart upload in progress for the following keys in your bucket, `example-bucket`.

`photos/2006/January/sample.jpg`

`photos/2006/February/sample.jpg`

`photos/2006/March/sample.jpg`

`videos/2006/March/sample.wmv`

`sample.jpg`

The following list multipart upload request specifies the delimiter parameter with value "/".

```
GET /?uploads&delimiter=/ HTTP/1.1
Host: example-bucket.s3.amazonaws.com
Date: Mon, 1 Nov 2010 20:34:56 GMT
Authorization: authorization string
```

The following sample response lists multipart uploads on the specified bucket, `example-bucket`.

The response returns multipart upload for the `sample.jpg` key in an <Upload> element.

However, because all the other keys contain the specified delimiter, a distinct substring, from the beginning of the key to the first occurrence of the delimiter, from each of these keys is returned in a <CommonPrefixes> element. The key substrings, `photos/` and `videos/`, in the <CommonPrefixes> element indicate that there are one or more in-progress multipart uploads with these key prefixes.

This is a useful scenario if you use key prefixes for your objects to create a logical folder like structure. In this case you can interpret the result as the folders `photos/` and `videos/` have one or more multipart uploads in progress.

```
<ListMultipartUploadsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Bucket>example-bucket</Bucket>
  <KeyMarker/>
  <UploadIdMarker/>
  <NextKeyMarker>sample.jpg</NextKeyMarker>
  <NextUploadIdMarker>Xgw4MJT6ZPAVx
pY0SAuGN7q4uWJJM22ZYg1W99trdp4tpO88.PT6.MhO0w2E17eutfAvQfQWoajgE_W2gpcxQw--
</NextUploadIdMarker>
  <Delimiter>/</Delimiter>
  <Prefix/>
  <MaxUploads>1000</MaxUploads>
  <IsTruncated>false</IsTruncated>
  <Upload>
    <Key>sample.jpg</Key>
   <UploadId>Agw4MJT6ZPAVxpY0SAuGN7q4uWJJM22ZYg1N99trdp4tpO88.PT6.MhO0w2E17eut
fAvQfQWoajgE_W2gpcxQw--</UploadId>
    <Initiator>
     <ID>314133b66967d86f031c7249d1d9a80249109428335cd0ef1cdc487b4566cb1b</ID>

      <DisplayName>s3-nickname</DisplayName>
    </Initiator>
    <Owner>
     <ID>314133b66967d86f031c7249d1d9a80249109428335cd0ef1cdc487b4566cb1b</ID>

      <DisplayName>s3-nickname</DisplayName>
    </Owner>
    <StorageClass>STANDARD</StorageClass>
    <Initiated>2010-11-26T19:24:17.000Z</Initiated>
  </Upload>
  <CommonPrefixes>
    <Prefix>photos/</Prefix>
  </CommonPrefixes>
  <CommonPrefixes>
    <Prefix>videos/</Prefix>
  </CommonPrefixes>
  </ListMultipartUploadsResult>
```

In addition to the delimiter parameter you can filter results by adding a `prefix` parameter as shown in the following request.

```
GET /?uploads&delimiter=/&prefix=photos/2006/ HTTP/1.1
Host: example-bucket.s3.amazonaws.com
Date: Mon, 1 Nov 2010 20:34:56 GMT
Authorization: authorization string
```

In this case the response will include only multipart uploads for keys that start with the specified prefix. The value returned in the <CommonPrefixes> element is a substring from the beginning of the key to the first occurrence of the specified delimiter after the prefix.

```
<?xml version="1.0" encoding="UTF-8"?>
<ListMultipartUploadsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Bucket>example-bucket</Bucket>
  <KeyMarker/>
  <UploadIdMarker/>
  <NextKeyMarker/>
  <NextUploadIdMarker/>
  <Delimiter>/</Delimiter>
  <Prefix>photos/2006/</Prefix>
  <MaxUploads>1000</MaxUploads>
  <IsTruncated>false</IsTruncated>
  <CommonPrefixes>
    <Prefix>photos/2006/February/</Prefix>
  </CommonPrefixes>
  <CommonPrefixes>
    <Prefix>photos/2006/January/</Prefix>
  </CommonPrefixes>
  <CommonPrefixes>
    <Prefix>photos/2006/March/</Prefix>
  </CommonPrefixes>
</ListMultipartUploadsResult>
```

Related Actions

- Initiate Multipart Upload (p. 282)
- Upload Part (p. 290)
- Complete Multipart Upload (p. 302)
- Abort Multipart Upload (p. 308)
- List Parts (p. 310)

PUT Bucket

Description

This implementation of the PUT operation creates a new bucket. To create a bucket, you must register with Amazon S3 and have a valid AWS Access Key ID to authenticate requests. Anonymous requests are never allowed to create buckets. By creating the bucket, you become the bucket owner.

Not every string is an acceptable bucket name. For information on bucket naming restrictions, see Working with Amazon S3 Buckets.

By default, the bucket is created in the US Standard region. You can optionally specify a region in the request body. You might choose a Region to optimize latency, minimize costs, or address regulatory requirements. For example, if you reside in Europe, you will probably find it advantageous to create buckets in the EU (Ireland) Region. For more information, see How to Select a Region for Your Buckets.

> **Note**
> If you create a bucket in a region other than US Standard, your application must be able to handle 307 redirect. For more information, go to Virtual Hosting of Buckets in *Amazon Simple Storage Service Developer Guide*.

When creating a bucket using this operation, you can optionally specify the accounts or groups that should be granted specific permissions on the bucket. There are two ways to grant the appropriate permissions using the request headers.

- Specify a canned ACL using the `x-amz-acl` request header. For more information, see Canned ACL in the *Amazon Simple Storage Service Developer Guide*.
- Specify access permissions explicitly using the `x-amz-grant-read`, `x-amz-grant-write`, `x-amz-grant-read-acp`, `x-amz-grant-write-acp`, `x-amz-grant-full-control` headers. These headers map to the set of permissions Amazon S3 supports in an ACL. For more information, go to Access Control List (ACL) Overview in the *Amazon Simple Storage Service Developer Guide*.

> **Note**
> You can use either a canned ACL or specify access permissions explicitly. You cannot do both.

Requests

Syntax

```
PUT / HTTP/1.1
Host: BucketName.s3.amazonaws.com
Content-Length: length
Date: date
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))

<CreateBucketConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <LocationConstraint>BucketRegion</LocationConstraint>
  </CreateBucketConfiguration>
```

> **Note**
> The syntax shows some of the request headers. For a complete list, see the Request Headers section.

Note
If you send your create bucket request to the `s3.amazonaws.com` endpoint, the request go to the `us-east-1` region. Accordingly, the signature calculations in Signature Version 4 must use `us-east-1` as region, even if the location constraint in the request specifies another region where the bucket is to be created.

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

This implementation of the operation can use the following request headers in addition to the request headers common to all operations. For more information, see Common Request Headers (p. 12).

When creating a bucket, you can grant permissions to individual AWS accounts or predefined groups defined by Amazon S3. This results in creation of the Access Control List (ACL) on the bucket. For more information, see Using ACLs. You have the following two ways to grant these permissions:

- **Specify a canned ACL —** Amazon S3 supports a set of predefined ACLs, known as canned ACLs. Each canned ACL has a predefined set of grantees and permissions. For more information, go to Canned ACL.

Name	Description	Required
x-amz-acl	The canned ACL to apply to the bucket you are creating. For more information, go to Canned ACL in the *Amazon Simple Storage Service Developer Guide*. Type: String Valid Values: `private | public-read | public-read-write | authenticated-read | bucket-owner-read | bucket-owner-full-control`	No

- **Specify access permissions explicitly —** If you want to explicitly grant access permissions to specific AWS accounts or groups, you use the following headers. Each of these headers maps to specific permissions Amazon S3 supports in an ACL. For more information, go to Access Control List (ACL) Overview. In the header value, you specify a list of grantees who get the specific permission

Name	Description	Required
x-amz-grant-read	Allows grantee to list the objects in the bucket. Type: String Default: None Constraints: None	No
x-amz-grant-write	Allows grantee to create, overwrite, and delete any object in the bucket. Type: String Default: None Constraints: None	No

Name	Description	Required
x-amz-grant-read-acp	Allows grantee to read the bucket ACL. Type: String Default: None Constraints: None	No
x-amz-grant-write-acp	Allows grantee to write the ACL for the applicable bucket. Type: String Default: None Constraints: None	No
x-amz-grant-full-control	Allows grantee the READ, WRITE, READ_ACP, and WRITE_ACP permissions on the bucket. Type: String Default: None Constraints: None	No

You specify each grantee as a `type=value` pair, where the type can be one of the following::

- **emailAddress —** if value specified is the email address of an AWS account
- **id —** if value specified is the canonical user ID of an AWS account
- **uri —** if granting permission to a predefined group.

For example, the following `x-amz-grant-read` header grants list objects permission to the AWS accounts identified by their email addresses.

```
x-amz-grant-read: emailAddress="xyz@amazon.com", emailAddress="abc@amazon.com"
```

For more information see, ACL Overview.

Request Elements

Name	Description	Required
CreateBucketConfiguration	Container for bucket configuration settings. Type: Container Ancestor: None	No

Name	Description	Required
LocationConstraint	Specifies the region where the bucket will be created. For more information about region endpoints and location constraints, go to Regions and Endpoints in the *Amazon Web Services Glossary*. Type: Enum Valid Values: EU | eu-west-1 | us-west-1 | us-west-2 | ap-southeast-1 | ap-southeast-2 | ap-northeast-1 | sa-east-1 | empty string (for the US Classic Region) Default: US Standard Ancestor: CreateBucketConfiguration	No

Response Elements

This implementation of the operation does not return response elements.

Special Errors

This implementation of the operation does not return special errors. For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

Sample Request

This request creates a bucket named `colorpictures`.

```
PUT / HTTP/1.1
Host: colorpictures.s3.amazonaws.com
Content-Length: 0
Date: Wed, 01 Mar  2006 12:00:00 GMT
Authorization: authorization string
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-id-2: YgIPIfBiKa2bj0KMg95r/0zo3emzU4dzsD4rcKCHQUAdQkf3ShJTOOpXUueF6QKo
x-amz-request-id: 236A8905248E5A01
Date: Wed, 01 Mar  2006 12:00:00 GMT

Location: /colorpictures
Content-Length: 0
Connection: close
Server: AmazonS3
```

Sample Request: Setting the region of a bucket

The following request sets the region the bucket to `EU`.

```
PUT / HTTP/1.1
Host: bucketName.s3.amazonaws.com
Date: Wed, 12 Oct 2009 17:50:00 GMT
Authorization: authorization string
Content-Type: text/plain
Content-Length: 124

<CreateBucketConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <LocationConstraint>EU</LocationConstraint>
</CreateBucketConfiguration >
```

Sample Response

Sample Request: Creating a bucket and configuring access permission using a canned ACL

This request creates a bucket named "colorpictures" and sets the ACL to `private`.

```
PUT / HTTP/1.1
Host: colorpictures.s3.amazonaws.com
Content-Length: 0
x-amz-acl: private
Date: Wed, 01 Mar  2006 12:00:00 GMT
Authorization: authorization string
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-id-2: YgIPIfBiKa2bj0KMg95r/0zo3emzU4dzsD4rcKCHQUAdQkf3ShJTOOpXUueF6QKo
x-amz-request-id: 236A8905248E5A01
Date: Wed, 01 Mar  2006 12:00:00 GMT

Location: /colorpictures
Content-Length: 0
Connection: close
Server: AmazonS3
```

Sample Request: Creating a bucket and configuring access permissions explicitly

This request creates a bucket named `colorpictures` and grants WRITE permission to the AWS account identified by an email address.

```
PUT HTTP/1.1
Host: colorpictures.s3.amazonaws.com
x-amz-date: Sat, 07 Apr 2012 00:54:40 GMT
Authorization: authorization string
x-amz-grant-write: emailAddress="xyz@amazon.com", emailAddress="abc@amazon.com"
```

Sample Response

```
HTTP/1.1 200 OK
```

Related Resources

- PUT Object (p. 250)
- DELETE Bucket (p. 69)

PUT Bucket acl

Description

This implementation of the PUT operation uses the *acl* subresource to set the permissions on an existing bucket using access control lists (ACL). For more information, go to Using ACLs. To set the ACL of a bucket, you must have WRITE_ACP permission.

You can use one of the following two ways to set a bucket's permissions:

- Specify the ACL in the request body
- Specify permissions using request headers

> **Note**
> You cannot specify access permission using both the body and the request headers.

Depending on your application needs, you may choose to set the ACL on a bucket using either the request body or the headers. For example, if you have an existing application that updates a bucket ACL using the request body, then you can continue to use that approach.

Requests

Syntax

The following request shows the syntax for sending the ACL in the request body. If you want to use headers to specify the permissions for the bucket, you cannot send the ACL in the request body. Instead, see Request Headers section for a list of headers you can use.

```
PUT /?acl HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))

<AccessControlPolicy>
  <Owner>
    <ID>ID</ID>
    <DisplayName>EmailAddress</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="CanonicalUser">
        <ID>ID</ID>
        <DisplayName>EmailAddress</DisplayName>
      </Grantee>
      <Permission>Permission</Permission>
    </Grant>
    ...
  </AccessControlList>
</AccessControlPolicy>
```

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

You can use the following request headers in addition to the Common Request Headers (p. 12).

These headers enable you to set access permissions using one of the following methods:

- Specify a canned ACL, or
- Specify the permission for each grantee explicitly

Amazon S3 supports a set of predefined ACLs, known as canned ACLs. Each canned ACL has a predefined set of grantees and permissions. For more information, see Canned ACL. To grant access permissions by specifying canned ACLs, you use the following header and specify the canned ACL name as its value. If you use this header, you cannot use other access control specific headers in your request.

Name	Description	Required
x-amz-acl	Sets the ACL of the bucket using the specified canned ACL. Type: String Valid Values: private | public-read | public-read-write | authenticated-read Default: private	No

If you need to grant individualized access permissions on a bucket, you can use the following "x-amz-grant-*permission*" headers. When using these headers you specify explicit access permissions and grantees (AWS accounts or a Amazon S3 groups) who will receive the permission. If you use these ACL specific headers, you cannot use *x-amz-acl* header to set a canned ACL.

> **Note**
> Each of the following request headers maps to specific permissions Amazon S3 supports in an ACL. For more information go to Access Control List (ACL) Overview.

Name	Description	Required
x-amz-grant-read	Allows the specified grantee(s) to list the objects in the bucket. Type: String Default: None Constraints: None	No
x-amz-grant-write	Allows the specified grantee(s) to create, overwrite, and delete any object in the bucket. Type: String Default: None Constraints: None	No
x-amz-grant-read-acp	Allows the specified grantee(s) to read the bucket ACL. Type: String Default: None Constraints: None	No

Name	Description	Required
x-amz-grant-write-acp	Allows the specified grantee(s) to write the ACL for the applicable bucket. Type: String Default: None Constraints: None	No
x-amz-grant-full-control	Allows the specified grantee(s) the READ, WRITE, READ_ACP, and WRITE_ACP permissions on the bucket. Type: String Default: None Constraints: None	No

For each of these headers, the value is a comma-separated list of one or more grantees. You specify each grantee as a `type=value` pair, where the `type` can be one of the following:

- **emailAddress —** if value specified is the email address of an AWS account
- **id —** if value specified is the canonical User ID of an AWS account
- **uri —** if granting permission to a predefined Amazon S3 group.

For example, the following `x-amz-grant-write` header grants create, overwrite, and delete objects permission to `LogDelivery` group predefined by Amazon S3 and two AWS accounts identified by their email addresses.

```
x-amz-grant-write: uri="http://acs.amazonaws.com/groups/s3/LogDelivery",
emailAddress="xyz@amazon.com", emailAddress="abc@amazon.com"
```

For more information, go to Access Control List (ACL) Overview. For more information about bucket logging, go to Server Access Logging.

Request Elements

If you decide to use the request body to specify an ACL, you must use the following elements.

Note
If you request the request body, you cannot use the request headers to set an ACL.

Name	Description	Required
`AccessControlList`	Container for Grant, Grantee, and Permission Type: Container Ancestors: `AccessControlPolicy`	No
`AccessControlPolicy`	Contains the elements that set the ACL permissions for an object per grantee. Type: String Ancestors: None	No
`DisplayName`	Screen name of the bucket owner. Type: String Ancestors: `AccessControlPolicy.Owner`	No

Name	Description	Required
Grant	Container for the grantee and his or her permissions. Type: Container Ancestors: `AccessControlPolicy.AccessControlList`	No
Grantee	The subject whose permissions are being set. For more information, see Grantee Values (p. 153). Type: String Ancestors: `AccessControlPolicy.AccessControlList.Grant`	No
ID	ID of the bucket owner, or the ID of the grantee. Type: String Ancestors: `AccessControlPolicy.Owner` | `AccessControlPolicy.AccessControlList.Grant`	No
Owner	Container for the bucket owner's display name and ID. Type: Container Ancestors: `AccessControlPolicy`	No
Permission	Specifies the permission given to the grantee. Type: String Valid Values: FULL_CONTROL | WRITE | WRITE_ACP | READ | READ_ACP Ancestors: `AccessControlPolicy.AccessControlList.Grant`	No

Grantee Values

You can specify the person (grantee) to whom you're assigning access rights (using request elements) in the following ways:

- By the person's ID:

```
<Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="Ca
nonicalUser"><ID>ID</ID><DisplayName>GranteesEmail</DisplayName>
</Grantee>
```

  DisplayName is optional and ignored in the request.

- By Email address:

```
<Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="AmazonCustomerByEmail"><EmailAddress>Grantees@email.com</EmailAd
dress>lt;/Grantee>
```

  The grantee is resolved to the *CanonicalUser* and, in a response to a `GET Object acl` request, appears as the *CanonicalUser*.

- By URI:

```
<Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="Group"><URI>http://acs.amazonaws.com/groups/global/Authenticated
Users</URI></Grantee>
```

Responses

Response Headers

The operation returns response header that are common to most responses. For more information, see Common Response Headers (p. 14).

Response Elements

This operation does not return response elements.

Special Errors

This operation does not return special errors. For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

Sample Request: Access permissions specified in the body

The following request grants access permission to the existing `examplebucket` bucket. The request specifies the ACL in the body. In addition to granting full control to the bucket owner, the XML specifies the following grants.

- Grant `AllUsers` group READ permission on the bucket.
- Grant the `LogDelivery` group WRITE permission on the bucket.
- Grant an AWS account, identified by email address, WRITE_ACP permission.
- Grant an AWS account, identified by canonical user ID, READ_ACP permission.

```
PUT ?acl HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Content-Length: 1660
x-amz-date: Thu, 12 Apr 2012 20:04:21 GMT
Authorization: authorization string

<AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Owner>
    <ID>852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID</ID>

    <DisplayName>OwnerDisplayName</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="CanonicalUser">
        <ID>852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonic
alUserID</ID>
        <DisplayName>OwnerDisplayName</DisplayName>
```

```
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="Group">
        <URI xmlns="">http://acs.amazonaws.com/groups/global/AllUsers</URI>
      </Grantee>
      <Permission xmlns="">READ</Permission>
    </Grant>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="Group">
        <URI xmlns="">http://acs.amazonaws.com/groups/s3/LogDelivery</URI>
      </Grantee>
      <Permission xmlns="">WRITE</Permission>
    </Grant>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="AmazonCustomerByEmail">
        <EmailAddress xmlns="">xyz@amazon.com</EmailAddress>
      </Grantee>
      <Permission xmlns="">WRITE_ACP</Permission>
    </Grant>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="CanonicalUser">
       <ID xmlns="">f30716ab7115dcb44a5ef76e9d74b8e20567f63TestAccountCanonic
alUserID</ID>
      </Grantee>
      <Permission xmlns="">READ_ACP</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-id-2: NxqO3PNiMHXXGwjgv15LLgUoAmPVmG0xtZw2sxePXLhpIvcyouXDrcQUaWWXcOK0
x-amz-request-id: C651BC9B4E1BD401
Date: Thu, 12 Apr 2012 20:04:28 GMT
Content-Length: 0
Server: AmazonS3
```

Sample Request: Access permissions specified using headers

The following request uses ACL-specific request headers to grant the following permissions:

- Write permission to the Amazon S3 `LogDelivery` group and an AWS account identified by the email xyz@amazon.com.
- Read permission to the Amazon S3 `AllUsers` group

```
PUT ?acl HTTP/1.1
Host: examplebucket.s3.amazonaws.com
```

```
x-amz-date: Sun, 29 Apr 2012 22:00:57 GMT
x-amz-grant-write: uri="http://acs.amazonaws.com/groups/s3/LogDelivery",
emailAddress="xyz@amazon.com"
x-amz-grant-read: uri="http://acs.amazonaws.com/groups/global/AllUsers"
Accept: */*
Authorization: authorization string
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-id-2: 0w9iImt23VF9s6QofOTDzelF7mrryz7d04Mw23FQCi4O205Zw28Zn+d340/RytoQ
x-amz-request-id: A6A8F01A38EC7138
Date: Sun, 29 Apr 2012 22:01:10 GMT
Content-Length: 0
Server: AmazonS3
```

Related Resources

- PUT Bucket (p. 144)
- DELETE Bucket (p. 69)
- GET Object ACL (p. 222)

PUT Bucket cors

Description

Sets the `cors` configuration for your bucket. If the configuration exists, Amazon S3 replaces it.

To use this operation, you must be allowed to perform the `s3:PutBucketCORS` action. By default, the bucket owner has this permission and can grant it to others.

You set this configuration on a bucket so that the bucket can service cross-origin requests. For example, you might want to enable a request whose origin is `http://www.example.com` to access your Amazon S3 bucket at `my.example.bucket.com` by using the browser's `XMLHttpRequest` capability.

To enable cross-origin resource sharing (CORS) on a bucket, you add the `cors` subresource to the bucket. The `cors` subresource is an XML document in which you configure rules that identify origins and the HTTP methods that can be executed on your bucket. The document is limited to 64 KB in size. For example, the following cors configuration on a bucket has two rules:

- The first `CORSRule` allows cross-origin PUT, POST and DELETE requests whose origin is `https://www.example.com` origins. The rule also allows all headers in a pre-flight OPTIONS request through the `Access-Control-Request-Headers` header. Therefore, in response to any pre-flight OPTIONS request, Amazon S3 will return any requested headers.
- The second rule allows cross-origin GET requests from all the origins. The '*' wildcard character refers to all origins.

```
<CORSConfiguration>
 <CORSRule>
   <AllowedOrigin>http://www.example.com</AllowedOrigin>

   <AllowedMethod>PUT</AllowedMethod>
   <AllowedMethod>POST</AllowedMethod>
   <AllowedMethod>DELETE</AllowedMethod>

   <AllowedHeader>*</AllowedHeader>
 </CORSRule>
 <CORSRule>
   <AllowedOrigin>*</AllowedOrigin>
   <AllowedMethod>GET</AllowedMethod>
 </CORSRule>
</CORSConfiguration>
```

The `cors` configuration also allows additional optional configuration parameters as shown in the following cors configuration on a bucket. For example, this `cors` configuration allows cross-origin PUT and POST requests from `http://www.example.com`.

```
<CORSConfiguration>
 <CORSRule>
   <AllowedOrigin>http://www.example.com</AllowedOrigin>
   <AllowedMethod>PUT</AllowedMethod>
   <AllowedMethod>POST</AllowedMethod>
   <AllowedMethod>DELETE</AllowedMethod>
   <AllowedHeader>*</AllowedHeader>
   <MaxAgeSeconds>3000</MaxAgeSeconds>
   <ExposeHeader>x-amz-server-side-encryption</ExposeHeader>
```

```
 </CORSRule>
</CORSConfiguration>
```

In the preceding configuration, `CORSRule` includes the following additional optional parameters:

- `MaxAgeSeconds`—Specifies the time in seconds that the browser will cache an Amazon S3 response to a pre-flight OPTIONS request for the specified resource. In this example, this parameter is 3000 seconds. Caching enables the browsers to avoid sending pre-flight OPTIONS request to Amazon S3 for repeated requests.
- `ExposeHeader`—Identifies the response header (in this case `x-amz-server-side-encryption`) that you want customers to be able to access from their applications (for example, from a JavaScript `XMLHttpRequest` object).

When Amazon S3 receives a cross-origin request (or a pre-flight OPTIONS request) against a bucket, it evaluates the `cors` configuration on the bucket and uses the first `CORSRule` rule that matches the incoming browser request to enable a cross-origin request. For a rule to match, the following conditions must be met:

- The request's `Origin` header must match `AllowedOrigin` elements.
- The request method (for example, GET, PUT, HEAD and so on) or the `Access-Control-Request-Method` header in case of a pre-flight `OPTIONS` request must be one of the `AllowedMethod` elements.
- Every header specified in the `Access-Control-Request-Headers` request header of a pre-flight request must match an `AllowedHeader` element.

For more information about CORS, go to Enabling Cross-Origin Resource Sharing in the *Amazon Simple Storage Service Developer Guide*.

Requests

Syntax

```
PUT /?cors HTTP/1.1
Host: bucketname.s3.amazonaws.com
Content-Length: length
Date: date
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))
Content-MD5: MD5

<CORSConfiguration>
  <CORSRule>
    <AllowedOrigin>Origin you want to allow cross-domain requests from</Allowe
dOrigin>
    <AllowedOrigin>...</AllowedOrigin>
    ...
    <AllowedMethod>HTTP method</AllowedMethod>
    <AllowedMethod>...</AllowedMethod>
    ...
    <MaxAgeSeconds>Time in seconds your browser to cache the pre-flight OPTIONS
 response for a resource</MaxAgeSeconds>
    <AllowedHeader>Headers that you want the browser to be allowed to
send</AllowedHeader>
    <AllowedHeader>...</AllowedHeader>
     ...
```

```
    <ExposeHeader>Headers in the response that you want accessible from client
 application</ExposeHeader>
    <ExposeHeader>...</ExposeHeader>
     ...
  </CORSRule>
  <CORSRule>
    ...
  </CORSRule>
    ...
</CORSConfiguration>
```

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

Name	Description	Required
Content-MD5	The base64-encoded 128-bit MD5 digest of the data. This header must be used as a message integrity check to verify that the request body was not corrupted in transit. For more information, go to RFC 1864. Type: String Default: None	Yes

Request Elements

Name	Description	Required
CORSConfiguration	Container for up to 100 `CORSRules` elements. Type: Container Children: `CORSRules` Ancestor: None	Yes
CORSRule	A set of origins and methods (cross-origin access that you want to allow). You can add up to 100 rules to the configuration. Type: Container Children: `AllowedOrigin`, `AllowedMethod`, `MaxAgeSeconds`, `ExposeHeader`, `ID`. Ancestor: `CORSConfiguration`	Yes
ID	A unique identifier for the rule. The ID value can be up to 255 characters long. The IDs help you find a rule in the configuration. Type: String Ancestor: `CORSRule`	No

Name	Description	Required
AllowedMethod	An HTTP method that you want to allow the origin to execute. Each *CORSRule* must identify at least one origin and one method. Type: Enum (GET, PUT, HEAD, POST, DELETE) Ancestor: CORSRule	Yes
AllowedOrigin	An origin that you want to allow cross-domain requests from. This can contain at most one * wild character. Each *CORSRule* must identify at least one origin and one method. The origin value can include at most one '*' wild character. For example, "http://*.example.com". You can also specify only * as the origin value allowing all origins cross-domain access. Type: String Ancestor: CORSRule	Yes
AllowedHeader	Specifies which headers are allowed in a pre-flight OPTIONS request via the Access-Control-Request-Headers header. Each header name specified in the Access-Control-Request-Headers header must have a corresponding entry in the rule. Amazon S3 will send only the allowed headers in a response that were requested. This can contain at most one * wild character. Type: String Ancestor: CORSRule	No
MaxAgeSeconds	The time in seconds that your browser is to cache the preflight response for the specified resource. A CORSRule can have at most one *MaxAgeSeconds* element. Type: Integer (seconds) Ancestor: CORSRule	No
ExposeHeader	One or more headers in the response that you want customers to be able to access from their applications (for example, from a JavaScript XMLHttpRequest object). You add one *ExposeHeader* element in the rule for each header. Type: String Ancestor: CORSRule	No

Responses

Response Headers

This implementation of the operation uses only response headers that are common to most responses. For more information, see Common Response Headers (p. 14).

Response Elements

This implementation of the operation does not return response elements.

Special Errors

This implementation of the operation does not return special errors. For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

The following examples add the `cors` subresource to a bucket.

Example : Configure cors

Sample Request

The following PUT request adds the `cors` subresource to a bucket (`examplebucket`).

```
PUT /?cors HTTP/1.1
Host: examplebucket.s3.amazonaws.com
x-amz-date: Tue, 21 Aug 2012 17:54:50 GMT
Content-MD5: 8dYiLewFWZyGgV2Q5FNI4W==
Authorization: authorization string
Content-Length: 216

<CORSConfiguration>
 <CORSRule>
   <AllowedOrigin>http://www.example.com</AllowedOrigin>
   <AllowedMethod>PUT</AllowedMethod>
   <AllowedMethod>POST</AllowedMethod>
   <AllowedMethod>DELETE</AllowedMethod>
   <AllowedHeader>*</AllowedHeader>
   <MaxAgeSeconds>3000</MaxAgeSec>
   <ExposeHeader>x-amz-server-side-encryption</ExposeHeader>
 </CORSRule>
 <CORSRule>
   <AllowedOrigin>*</AllowedOrigin>
   <AllowedMethod>GET</AllowedMethod>
   <AllowedHeader>*</AllowedHeader>
   <MaxAgeSeconds>3000</MaxAgeSeconds>
 </CORSRule>
</CORSConfiguration>
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-id-2: CCshOvbOPfxzhwOADyC4qHj/Ck3F9Q0viXKw3rivZ+GcBoZSOOahvEJfPisZB7B
x-amz-request-id: BDC4B83DF5096BBE
Date: Tue, 21 Aug 2012 17:54:50 GMT
Server: AmazonS3
```

Related Resources

- GET Bucket cors (p. 92)
- DELETE Bucket cors (p. 71)
- OPTIONS object (p. 235)

PUT Bucket lifecycle

Description

Creates a new lifecycle configuration for the bucket or replaces an existing lifecycle configuration. For information about lifecycle configuration, go to Object Lifecycle Management in the *Amazon Simple Storage Service Developer Guide*.

Permissions

By default, all Amazon S3 resources are private, including buckets, objects, and related subresources (for example, lifecycle configuration and website configuration). Only the resource owner, an AWS account that created it, can access the resource. The resource owner can optionally grant access permissions to others by writing an access policy. For this operation, a user must get the `s3:PutLifecycleConfiguration` permission.

You can also explicitly deny permissions. Explicit deny also supersedes any other permissions. If you want to block users or accounts from removing or deleting objects from your bucket, you must deny them permissions for the following actions.

- `s3:DeleteObject`
- `s3:DeleteObjectVersion`
- `s3:PutLifecycleConfiguration`

For more information about permissions, go to Managing Access Permissions to Your Amazon S3 Resources section in the *Amazon Simple Storage Service Developer Guide*.

Requests

Syntax

```
PUT /?lifecycle HTTP/1.1
Host: bucketname.s3.amazonaws.com
Content-Length: length
Date: date
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))
Content-MD5: MD5

Lifecycle configuration in the request body
```

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

Name	Description	Required
Content-MD5	The base64-encoded 128-bit MD5 digest of the data. This header must be used as a message integrity check to verify that the request body was not corrupted in transit. For more information, go to RFC 1864. Type: String Default: None	Yes

Request Body

In the request, you specify lifecycle configuration in the request body. The lifecycle configuration is specified as XML. The following is an introductory example lifecycle configuration skeleton. It specifies one rule. The Prefix in the rule identifies objects to which the rule applies. The rule also specifies two actions (`Transition`and `Expiration`). Each action specifies a timeline when you want Amazon S3 to perform the action. The Status indicates whether the rule is enabled or disabled.

```
<LifecycleConfiguration>
    <Rule>
        <ID>sample-rule</ID>
        <Prefix>key-prefix</Prefix>
        <Status>rule-status</Status>
        <Transition>
           <Date>value</Date>
           <StorageClass>GLACIER</StorageClass>
        </Transition>
        <Expiration>
           <Days>value</Days>
        </Expiration>
    </Rule>
</LifecycleConfiguration>
```

If the state of your bucket is versioning-enabled or versioning-suspended, you can have many versions of the same object, one current version, and zero or more noncurrent versions. The following lifecycle configuration specifies the actions (`NoncurrentVersionTransition`, `NoncurrentVersionExpiration`) that are specific to noncurrent object versions.

```
<LifecycleConfiguration>
    <Rule>
        <ID>sample-rule</ID>
        <Prefix>key-prefix</Prefix>
        <Status>rule-status</Status>
        <NoncurrentVersionTransition>
           <NoncurrentDays>value</NoncurrentDays>
           <StorageClass>GLACIER</StorageClass>
        </NoncurrentVersionTransition>
        <NoncurrentVersionExpiration>
           <NoncurrentDays>value</NoncurrentDays>
        </NoncurrentVersionExpiration>
```

```
    </Rule>
</LifecycleConfiguration>
```

The following table describes the XML elements in the lifecycle configuration:

Name	Description	Required
Date	Specifies the date after which you want the corresponding action to take effect. When the action is in effect, Amazon S3 will perform the specific action on the applicable objects as they appear in the bucket (you identify applicable objects in the lifecycle `Rule` in which the action is defined). For example, suppose you add a `Transition` action to take effect on December 31, 2014. Suppose this action applies to objects with key prefix "documents/". When the action takes effect on this date, Amazon S3 transitions existing applicable objects to the GLACIER storage class. As long as the action is in effects, Amazon S3 will transition any new objects, even after December 31, 2014. The date value must conform to the ISO 8601 format. The time is always midnight UTC. Type: String Ancestor: `Expiration` or `Transition`	Yes, if `Days` is absent.
Days	Specifies the number of days after object creation when the specific rule action takes effect. Type: Nonnegative Integer when used with `Transition`, Positive Integer when used with `Expiration`. Ancestor: `Expiration`, `Transition`.	Yes, if `Date` is absent.

Name	Description	Required
Expiration	This action specifies a period in an object's lifetime when Amazon S3 should take the appropriate expiration action. The action Amazon S3 takes depends on whether the bucket is versioning-enabled. • If versioning has never been enabled on the bucket, Amazon S3 deletes the only copy of the object permanently. • Otherwise, if your bucket is versioning-enabled (or versioning is suspended), the action applies only to the current version of the object. A versioning-enabled bucket can have many versions of the same object, one current version, and zero or more noncurrent versions. Instead of deleting the current version, Amazon S3 makes it a noncurrent version by adding a delete marker as the new current version. **Important** If your bucket state is versioning-suspended Amazon S3 creates a delete marker with version ID `null`. If you have a version with version ID `null`, then Amazon S3 overwrites that version. **Note** To set expiration for noncurrent objects, you must use the *NoncurrentVersionExpiration* action. Type: Container Children: Days or Date Ancestor: Rule	Yes, if no other action is present in the `Rule`.
ID	Unique identifier for the rule. The value cannot be longer than 255 characters. Type: String Ancestor: Rule	No
LifecycleConfiguration	Container for lifecycle rules. You can add as many as 1000 rules. Type: Container Children: Rule Ancestor: None	Yes

Name	Description	Required
NoncurrentDays	Specifies the number of days an object is noncurrent before Amazon S3 can perform the associated action. For information about the noncurrent days calculations, see How Amazon S3 Calculates When an Object Became Noncurrent in the *Amazon Simple Storage Service Developer Guide*. Type: Nonnegative Integer when used with `NoncurrentVersionTransition`, Positive Integer when used with `NoncurrentVersionExpiration`. Ancestor: `NoncurrentVersionExpiration` or `NoncurrentVersionTransition`	Yes
NoncurrentVersionExpiration	Specifies when noncurrent object versions expire. Upon expiration, Amazon S3 permanently deletes the noncurrent object versions. You set this lifecycle configuration action on a bucket that has versioning enabled (or suspended) to request that Amazon S3 delete noncurrent object versions at a specific period in the object's lifetime. Type: Container Children: NoncurrentDays Ancestor: Rule	Yes, if no other action is present in the `Rule`.
NoncurrentVersionTransition	Container for the transition rule that describes when noncurrent objects transition to the GLACIER storage class. If your bucket is versioning-enabled (or versioning is suspended), you can set this action to request that Amazon S3 transition noncurrent object versions to the GLACIER storage class at a specific period in the object's lifetime. Type: Container Children: NoncurrentDays and StorageClass Ancestor: Rule	Yes, if no other action is present in the `Rule`.
Prefix	Object key prefix identifying one or more objects to which the rule applies. Type: String Ancestor: Rule	Yes
Rule	Container for a lifecycle rule. A lifecycle configuration can contain as many as 1000 rules. Type: Container Ancestor:LifecycleConfiguration	Yes

Name	Description	Required
Status	If Enabled, Amazon S3 executes the rule as scheduled. If Disabled, Amazon S3 ignores the rule. Type: String Ancestor: Rule Valid values: Enabled, Disabled.	Yes
StorageClass	Specifies the Amazon S3 storage class to which you want the object to transition. Type: String Ancestor: Transition and NoncurrentVersionTransition Valid values: GLACIER.	Yes
Transition	This action specifies a period in the objects' lifetime when Amazon S3 should transition them to the GLACIER storage class. When this action is in effect, what Amazon S3 does depends on whether the bucket is versioning-enabled. • If versioning has never been enabled on the bucket, Amazon S3 transitions the only copy of the object to the GLACIER storage class. • Otherwise, when your bucket is versioning-enabled (or versioning is suspended) Amazon S3 transitions only the current versions of objects identified in the rule. **Note** A versioning-enabled bucket can have many versions of an object. This action has no impact on the noncurrent object versions. To transition noncurrent objects to the GLACIER storage class, you must use the *NoncurrentVersionTransition* action. Type: Container Children: Days or Date, and StorageClass Ancestor: Rule	Yes, if no other action is present in the `Rule`.

Responses

Response Headers

This implementation of the operation uses only response headers that are common to most responses. For more information, see Common Response Headers (p. 14).

Response Elements

This implementation of the operation does not return response elements.

Special Errors

This implementation of the operation does not return special errors. For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

Example 1: Add lifecycle configuration - bucket not versioning-enabled

The following lifecycle configuration specifies two rules, each with one action.

- The Transition action requests Amazon S3 to transition objects with the "documents/" prefix to the GLACIER storage class 30 days after creation.
- The Expiration action requests Amazon S3 to delete objects with the "logs/" prefix 365 days after creation.

```
<LifecycleConfiguration>
  <Rule>
    <ID>id1</ID>
    <Prefix>documents/</Prefix>
    <Status>Enabled</Status>
    <Transition>
      <Days>30</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
  </Rule>
  <Rule>
    <ID>id2</ID>
    <Prefix>logs/</Prefix>
    <Status>Enabled</Status>
    <Expiration>
      <Days>365</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

The following is a sample `PUT /?lifecycle` request that adds the preceding lifecycle configuration to the `examplebucket` bucket.

```
PUT /?lifecycle HTTP/1.1
Host: examplebucket.s3.amazonaws.com
x-amz-date: Wed, 14 May 2014 02:11:21 GMT
Content-MD5: q6yJDlIkcBaGGfb3QLY69A==
Authorization: authorization string
Content-Length: 415

<LifecycleConfiguration>
  <Rule>
    <ID>id1</ID>
    <Prefix>documents/</Prefix>
    <Status>Enabled</Status>
```

```
    <Transition>
      <Days>30</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
  </Rule>
  <Rule>
    <ID>id2</ID>
    <Prefix>logs/</Prefix>
    <Status>Enabled</Status>
    <Expiration>
      <Days>365</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

The following is a sample response.

```
HTTP/1.1 200 OK
x-amz-id-2: r+qR7+nhXtJDDIJ0JJYcd+1j5nM/rUFiiiZ/fNbDOsd3JUE8NWMLNHXmvPfwMpdc
x-amz-request-id: 9E26D08072A8EF9E
Date: Wed, 14 May 2014 02:11:22 GMT
Content-Length: 0
Server: AmazonS3
```

Example 2: Add lifecycle configuration - bucket is versioning-enabled

The following lifecycle configuration specifies two rules, each with one action for Amazon S3 to perform. You specify these actions when your bucket is versioning-enabled or versioning is suspended:

- The `NoncurrentVersionExpiration` action requests Amazon S3 to expire noncurrent versions of objects with the "logs/" prefix 100 days after the objects become noncurrent.
- The `NoncurrentVersionTransition` action requests Amazon S3 to transition noncurrent versions of objects with the "documents/" prefix to the GLACIER storage class 30 days after they become noncurrent.

```
<LifeCycleConfiguration>
  <Rule>
    <ID>DeleteAfterBecomingNonCurrent</ID>
    <Prefix>logs/</Prefix>
    <Status>Enabled</Status>
    <NoncurrentVersionExpiration>
      <NoncurrentDays>100</NoncurrentDays>
    </NoncurrentVersionExpiration>
  </Rule>
  <Rule>
    <ID>TransitionAfterBecomingNonCurrent</ID>
    <Prefix>documents/</Prefix>
    <Status>Enabled</Status>
    <NoncurrentVersionTransition>
      <NoncurrentDays>30</NoncurrentDays>
      <StorageClass>GLACIER</StorageClass>
    </NoncurrentVersionTransition>
  </Rule>
</LifeCycleConfiguration>
```

The following is a sample `PUT /?lifecycle` request that adds the preceding lifecycle configuration to the `examplebucket` bucket.

```
PUT /?lifecycle HTTP/1.1
Host: examplebucket.s3.amazonaws.com
x-amz-date: Wed, 14 May 2014 02:21:48 GMT
Content-MD5: 96rxH9mDqVNKkaZDddgnw==
Authorization: authorization string
Content-Length: 598

<LifeCycleConfiguration>
  <Rule>
    <ID>DeleteAfterBecomingNonCurrent</ID>
    <Prefix>logs/</Prefix>
    <Status>Enabled</Status>
    <NoncurrentVersionExpiration>
      <NoncurrentDays>1</NoncurrentDays>
    </NoncurrentVersionExpiration>
  </Rule>
  <Rule>
    <ID>TransitionImmediatelyAfterBecomingNonCurrent</ID>
    <Prefix>documents/</Prefix>
    <Status>Enabled</Status>
    <NoncurrentVersionTransition>
      <NoncurrentDays>0</NoncurrentDays>
      <StorageClass>GLACIER</StorageClass>
    </NoncurrentVersionTransition>
  </Rule>
</LifeCycleConfiguration>
```

The following is a sample response.

```
HTTP/1.1 200 OK
x-amz-id-2: aXQ+KbIrmMmoO//3bMdDTw/CnjArwje+J49Hf+j44yRb/VmbIk
gIO5A+PT98Cp/6k07hf+LD2mY=
x-amz-request-id: 02D7EC4C10381EB1
Date: Wed, 14 May 2014 02:21:50 GMT
Content-Length: 0
Server: AmazonS3
```

Related Resources

- GET Bucket lifecycle (p. 95)
- POST Object restore (p. 247)
- By default, a resource owner, in this case a bucket owner (the AWS account that created the bucket), can perform any of the operations, and can also grant others permission to perform the operation. For more information, see the following topics in the *Amazon Simple Storage Service Developer Guide*.
  - Specifying Permissions in a Policy
  - Managing Access Permissions to Your Amazon S3 Resources

PUT Bucket policy

Description

This implementation of the PUT operation uses the *policy* subresource to add to or replace a policy on a bucket. If the bucket already has a policy, the one in this request completely replaces it. To perform this operation, you must be the bucket owner.

If you are not the bucket owner but have *PutBucketPolicy* permissions on the bucket, Amazon S3 returns a *405 Method Not Allowed*. In all other cases for a PUT bucket policy request that is not from the bucket owner, Amazon S3 returns 403 Access Denied. There are restrictions about who can create bucket policies and which objects in a bucket they can apply to. For more information, go to Using Bucket Policies.

Requests

Syntax

```
PUT /?policy HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))

Policy written in JSON
```

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

This implementation of the operation uses only request headers that are common to all operations. For more information, see Common Request Headers (p. 12).

Request Elements

The body is a JSON string containing the policy contents containing the policy statements.

Responses

Response Headers

This implementation of the operation uses only response headers that are common to most responses. For more information, see Common Response Headers (p. 14).

Response Elements

PUT response elements return whether the operation succeeded or not.

Special Errors

This implementation of the operation does not return special errors. For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

Sample Request

The following request shows the PUT individual policy request for the bucket.

```
PUT /?policy HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Tue, 04 Apr 2010 20:34:56 GMT
Authorization: authorization string

{
"Version":"2008-10-17",
"Id":"aaaa-bbbb-cccc-dddd",
"Statement" : [
    {
        "Effect":"Allow",
        "Sid":"1",
        "Principal" : {
            "AWS":["111122223333","444455556666"]
        },
        "Action":["s3:*"],
        "Resource":"arn:aws:s3:::bucket/*"
    }
 ]
}
```

Sample Response

```
HTTP/1.1 204 No Content
x-amz-id-2: Uuag1LuByR5Onimru9SAMPLEAtRPfTaOFg==
x-amz-request-id: 656c76696e6727732SAMPLE7374
Date: Tue, 04 Apr 2010 20:34:56 GMT
Connection: keep-alive
Server: AmazonS3
```

Related Resources

- PUT Bucket (p. 144)
- DELETE Bucket (p. 69)

PUT Bucket logging

Description

Note
The logging implementation of PUT Bucket is a beta feature.

This implementation of the PUT operation uses the *logging* subresource to set the logging parameters for a bucket and to specify permissions for who can view and modify the logging parameters. To set the logging status of a bucket, you must be the bucket owner.

The bucket owner is automatically granted FULL_CONTROL to all logs. You use the *Grantee* request element to grant access to other people. The *Permissions* request element specifies the kind of access the grantee has to the logs.

To enable logging, you use *LoggingEnabled* and its children request elements.

To disable logging, you use an empty *BucketLoggingStatus* request element:

```
<BucketLoggingStatus xmlns="http://doc.s3.amazonaws.com/2006-03-01" />
```

For more information about creating a bucket, see PUT Bucket (p. 144). For more information about returning the logging status of a bucket, see GET Bucket logging (p. 105).

Requests

Syntax

```
PUT /?logging HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))

Request elements vary depending on what you're setting.
```

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

This implementation of the operation uses only request headers that are common to all operations. For more information, see Common Request Headers (p. 12).

Request Elements

Name	Description	Required
BucketLoggingStatus	Container for logging status information. Type: Container Children: LoggingEnabled Ancestry: None	Yes

Name	Description	Required
EmailAddress	E-mail address of the person being granted logging permissions. Type: String Children: None Ancestry: BucketLoggingStatus.LoggingEnabled.TargetGrants.Grant.Grantee	No
Grant	Container for the grantee and his/her logging permissions. Type: Container Children: Grantee, Permission Ancestry: BucketLoggingStatus.LoggingEnabled.TargetGrants	No
Grantee	Container for *EmailAddress* of the person being granted logging permissions. For more information, see Grantee Values (p. 175). Type: Container Children: EmailAddress Ancestry: BucketLoggingStatus.LoggingEnabled.TargetGrants.Grant	No
LoggingEnabled	Container for logging information. This element is present when you are enabling logging (and not present when you are disabling logging). Type: Container Children: Grant, TargetBucket, TargetPrefix Ancestry: BucketLoggingStatus	No
Permission	Logging permissions given to the *Grantee* for the bucket. The bucket owner is automatically granted FULL_CONTROL to all logs delivered to the bucket. This optional element enables you grant access to others. Type: String Valid Values: FULL_CONTROL | READ | WRITE Children: None Ancestry: BucketLoggingStatus.LoggingEnabled.TargetGrants.Grant	No
TargetBucket	Specifies the bucket where you want Amazon S3 to store server access logs. You can have your logs delivered to any bucket that you own, including the same bucket that is being logged. You can also configure multiple buckets to deliver their logs to the same target bucket. In this case you should choose a different TargetPrefix for each source bucket so that the delivered log files can be distinguished by key. Type: String Children: None Ancestry: BucketLoggingStatus.LoggingEnabled	No

Name	Description	Required
TargetGrants	Container for granting information. Type: Container Children: Grant, Permission Ancestry: BucketLoggingStatus.LoggingEnabled	No
TargetPrefix	This element lets you specify a prefix for the keys that the log files will be stored under. Type: String Children: None Ancestry: BucketLoggingStatus.LoggingEnabled	No

Grantee Values

You can specify the person (grantee) to whom you're assigning access rights (using request elements) in the following ways:

- By the person's ID:

```
<Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="Ca
nonicalUser"><ID>ID</ID><DisplayName>GranteesEmail</DisplayName>
</Grantee>
```

  DisplayName is optional and ignored in the request.
- By Email address:

```
<Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="AmazonCustomerByEmail"><EmailAddress>Grantees@email.com</EmailAd
dress>lt;/Grantee>
```

  The grantee is resolved to the *CanonicalUser* and, in a response to a `GET Object acl` request, appears as the *CanonicalUser*.
- By URI:

```
<Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="Group"><URI>http://acs.amazonaws.com/groups/global/Authenticated
Users</URI></Grantee>
```

Responses

Response Headers

This implementation of the operation uses only response headers that are common to most responses. For more information, see Common Response Headers (p. 14).

Response Elements

This implementation of the operation does not return response elements.

Special Errors

This implementation of the operation does not return special errors. For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

Sample Request

This request enables logging and gives the grantee of the bucket READ access to the logs.

```
PUT ?logging HTTP/1.1
Host: quotes.s3.amazonaws.com
Content-Length: 214
Date: Wed, 25 Nov 2009 12:00:00 GMT
Authorization: authorization string

<?xml version="1.0" encoding="UTF-8"?>
<BucketLoggingStatus xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <LoggingEnabled>
    <TargetBucket>mybucketlogs</TargetBucket>
    <TargetPrefix>mybucket-access_log-/</TargetPrefix>
    <TargetGrants>
      <Grant>
        <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:type="AmazonCustomerByEmail">
          <EmailAddress>user@company.com</EmailAddress>
        </Grantee>
        <Permission>READ</Permission>
      </Grant>
    </TargetGrants>
  </LoggingEnabled>
</BucketLoggingStatus>
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-id-2: YgIPIfBiKa2bj0KMg95r/0zo3emzU4dzsD4rcKCHQUAdQkf3ShJTOOpXUueF6QKo
x-amz-request-id: 236A8905248E5A01
Date: Wed, 01 Mar  2006 12:00:00 GMT
```

Sample Request Disabling Logging

This request disables logging on the bucket, `quotes`.

```
PUT ?logging HTTP/1.1
Host: quotes.s3.amazonaws.com
Content-Length: 214
Date: Wed, 25 Nov 2009 12:00:00 GMT
Authorization: authorization string

<?xml version="1.0" encoding="UTF-8"?>
<BucketLoggingStatus xmlns="http://doc.s3.amazonaws.com/2006-03-01" />
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-id-2: YgIPIfBiKa2bj0KMg95r/0zo3emzU4dzsD4rcKCHQUAdQkf3ShJTOOpXUueF6QKo
x-amz-request-id: 236A8905248E5A01
Date: Wed, 01 Mar  2006 12:00:00 GMT
```

Related Resources

- PUT Object (p. 250)
- DELETE Bucket (p. 69)
- PUT Bucket (p. 144)
- GET Bucket logging (p. 105)

PUT Bucket notification

Description

This implementation of the PUT operation uses the *notification* subresource to enable notifications of specified events for a bucket. Currently, the *s3:ReducedRedundancyLostObject* event is the only event supported for notifications. The *s3:ReducedRedundancyLostObject* event is triggered when Amazon S3 detects that it has lost all replicas of an object and can no longer service requests for that object.

If the bucket owner and Amazon SNS topic owner are the same, the bucket owner has permission to publish notifications to the topic by default. Otherwise, the owner of the topic must create a policy to enable the bucket owner to publish to the topic. For more information about creating this policy, go to Example Cases for Amazon SNS Access Control.

By default, only the bucket owner can configure notifications on a bucket. However, bucket owners can use a bucket policy to grant permission to other users to set this configuration with *s3:PutBucketNotification* permission.

After you call the PUT operation to configure notifications on a bucket, Amazon S3 publishes a test notification to ensure that the topic exists and that the bucket owner has permission to publish to the specified topic. If the notification is successfully published to the SNS topic, the PUT operation updates the bucket configuration and returns the 200 OK response with a *x-amz-sns-test-message-id* header containing the message ID of the test notification sent to topic.

To turn off notifications on a bucket, you specify an empty *NotificationConfiguration* element in your request: `<NotificationConfiguration />`

For more information about setting and reading the notification configuration on a bucket, see Setting Up Notification of Bucket Events. For more information about bucket policies, see Using Bucket Policies.

Requests

Syntax

```
PUT /?notification HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))
<NotificationConfiguration>
   <TopicConfiguration>
       <Topic>TopicARN</Topic>
       <Event>Event</Event>
   </TopicConfiguration>
</NotificationConfiguration>
```

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

This implementation of the operation uses only request headers that are common to all operations. For more information, see Common Request Headers (p. 12).

Request Elements

Name	Description	Required
NotificationConfiguration	Container for specifying the notification configuration of the bucket. If this element is empty, notifications are turned off on the bucket. Type: Container Children: *TopicConfiguration* Ancestry: None	Yes
TopicConfiguration	Container for specifying the topic configuration for the notification. Currently, only one topic can be configured for notifications. Type: Container Children: *Topic*, *Event* Ancestry: *NotificationConfiguration*	No
Topic	Amazon SNS topic to which Amazon S3 will publish a message to report the specified events for the bucket. Type: String Ancestry: *TopicConfiguration*	No
Event	Bucket event for which to send notifications. Currently, *s3:ReducedRedundancyLostObject* is the only event supported for notifications. Type: String Valid Values: *s3:ReducedRedundancyLostObject* Ancestry: *TopicConfiguration*	No

Responses

Response Headers

This implementation of the operation uses only response headers that are common to most responses. For more information, see Common Response Headers (p. 14).

Response Elements

This implementation of the operation does not return response elements.

Special Errors

Amazon S3 checks the validity of the proposed *NotificationConfiguration* element and verifies whether the proposed configuration is valid when you call the PUT operation. The following table lists the errors and possible causes.

HTTP Error	Code	Cause
`HTTP 400 Bad Request`	InvalidArgument	The following conditions can cause this error: • The specified event is not supported for notifications. • The specified topic ARN does not exist or is not well-formed. Verify the topic ARN. • The specified topic is in a different region than the bucket. You must use a topic that resides in the same Region as the bucket. • The bucket owner does not have *`Publish`* permission on the specified topic.
`HTTP 403 Forbidden`	AccessDenied	You are not the owner of the specified bucket or you do not have the *`s3:PutBucketNotification`* bucket permission to set the notification configuration on the bucket.

For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

Sample Requests

This request enables notification on bucket `quotes.s3.amazonaws.com` for the event *`s3:ReducedRedundancyLostObject`* with notifications published to the topic arn:aws:sns:us-east-1:123456789012:myTopic.

```
PUT ?notification HTTP/1.1
Host: quotes.s3.amazonaws.com
Date: Wed, 02 June 2010 12:00:00 GMT
Authorization: authorization string
<NotificationConfiguration>
   <TopicConfiguration>
       <Topic>arn:aws:sns:us-east-1:123456789012:myTopic</Topic>
       <Event>s3:ReducedRedundancyLostObject</Event>
   </TopicConfiguration>
</NotificationConfiguration>
```

This request turns off notification on the `quotes.s3.amazonaws.com` bucket.

```
PUT ?notification HTTP/1.1
Host: quotes.s3.amazonaws.com
Date: Wed, 02 June 2010 12:01:00 GMT
Authorization: authorization string
<NotificationConfiguration />
```

Sample Responses

In this response, you are notified that the notification configuration was successful. It also returns the ID of the test message Amazon S3 sent to the topic.

```
HTTP/1.1 200 OK
x-amz-id-2: YgIPIfBiKa2bj0KMgUAdQkf3ShJTOOpXUueF6QKo
x-amz-request-id: 236A8905248E5A01
x-amz-sns-test-message-id: feeb1dff-cc96-449d-964c-f8a1890fd007
Date: Wed, 02 June 2010 12:00:00 GMT
Content-Length: 0
Connection: close
Server: AmazonS3
```

This response returns that the notification was turned off successfully. Note that Amazon S3 doesn't send a test notification when notifications are turned off.

```
HTTP/1.1 200 OK
x-amz-id-2: YgIPIfBiKa2bj0KMgUAdQkf3ShJTOOpXUueF6QKo
x-amz-request-id: 236A890524860101
Date: Wed, 02 June 2010 12:01:00 GMT
Content-Length: 0
Connection: close
Server: AmazonS3
```

Related Resources

- GET Bucket notification (p. 108)

PUT Bucket tagging

Description

This implementation of the PUT operation uses the *tagging* subresource to add a set of tags to an existing bucket.

Use tags to organize your AWS bill to reflect your own cost structure. To do this, sign up to get your AWS account bill with tag key values included. Then, to see the cost of combined resources, organize your billing information according to resources with the same tag key values. For example, you can tag several resources with a specific application name, and then organize your billing information to see the total cost of that application across several services. For more information, see Cost Allocation and Tagging in *About AWS Billing and Cost Management*.

To use this operation, you must have permission to perform the `s3:PutBucketTagging` action. By default, the bucket owner has this permission and can grant this permission to others.

Requests

Syntax

The following request shows the syntax for sending tagging information in the request body.

```
PUT /?tagging HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))

<Tagging>
  <TagSet>
     <Tag>
       <Key>Tag Name</Key>
       <Value>Tag Value</Value>
     </Tag>
  </TagSet>
</Tagging>
```

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

Content-MD5 will be a required header for this operation.

Request Elements

Name	Description	Required
Tagging	Container for the `TagSet` and `Tag` elements. Type: String Ancestors: None	Yes

Name	Description	Required
TagSet	Container for a set of tags Type: Container Ancestors: `Tagging`	Yes
Tag	Container for tag information. Type: Container Ancestors: `TagSet`	Yes
Key	Name of the tag. Type: String Ancestors: `Tag`	Yes
Value	Value of the tag. Type: String Ancestors: `Tag`	Yes

Responses

Response Headers

The operation returns response header that are common to most responses. For more information, see Common Response Headers (p. 14).

Response Elements

This operation does not return response elements.

Special Errors

- InvalidTagError - The tag provided was not a valid tag. This error can occur if the tag did not pass input validation. See the CostAllocation docs for a description of valid tags.
- MalformedXMLError - The XML provided does not match the schema.
- OperationAbortedError - A conflicting conditional operation is currently in progress against this resource. Please try again.
- InternalError - The service was unable to apply the provided tag to the bucket.

Examples

Sample Request: Add tag set to a bucket

The following request adds a tag set to the existing `examplebucket` bucket.

```
PUT ?tagging HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Content-Length: 1660
x-amz-date: Thu, 12 Apr 2012 20:04:21 GMT
Authorization: authorization string

<Tagging>
```

```
  <TagSet>
    <Tag>
      <Key>Project</Key>
      <Value>Project One</Value>
    </Tag>
    <Tag>
      <Key>User</Key>
      <Value>jsmith</Value>
    </Tag>
  </TagSet>
</Tagging>
```

Sample Response

```
HTTP/1.1 204 No Content
x-amz-id-2: YgIPIfBiKa2bj0KMgUAdQkf3ShJTOOpXUueF6QKo
x-amz-request-id: 236A8905248E5A01
Date: Wed, 01 Oct  2012 12:00:00 GMT
```

Related Resources

- GET Bucket tagging (p. 111)
- DELETE Bucket tagging (p. 77)

PUT Bucket requestPayment

Description

This implementation of the PUT operation uses the *requestPayment* subresource to set the request payment configuration of a bucket. By default, the bucket owner pays for downloads from the bucket. This configuration parameter enables the bucket owner (only) to specify that the person requesting the download will be charged for the download. For more information, see Requester Pays Buckets.

Requests

Syntax

```
PUT ?requestPayment HTTP/1.1
Host: BucketName.s3.amazonaws.com
Content-Length: length
Date: date
Authorization:signatureValue

<RequestPaymentConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Payer>payer</Payer>
</RequestPaymentConfiguration>
```

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

This implementation of the operation uses only request headers that are common to all operations. For more information, see Common Request Headers (p. 12).

Request Elements

Name	Description
Payer	Specifies who pays for the download and request fees. Type: Enum Valid Values: Requester | BucketOwner Ancestor: RequestPaymentConfiguration
RequestPaymentConfiguration	Container for *Payer*. Type: Container

Responses

Response Headers

This implementation of the operation uses only response headers that are common to most responses. For more information, see Common Response Headers (p. 14).

Response Elements

This implementation of the operation does not return response elements.

Special Errors

This implementation of the operation does not return special errors. For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

Sample Request

This request creates a Requester Pays bucket named "colorpictures."

```
PUT ?requestPayment HTTP/1.1
Host: colorpictures.s3.amazonaws.com
Content-Length: 173
Date: Wed, 01 Mar  2006 12:00:00 GMT
Authorization: authorization string

<RequestPaymentConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Payer>Requester</Payer>
</RequestPaymentConfiguration>
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-id-2: YgIPIfBiKa2bj0KMg95r/0zo3emzU4dzsD4rcKCHQUAdQkf3ShJTOOpXUueF6QKo
x-amz-request-id: 236A8905248E5A01
Date: Wed, 01 Mar  2006 12:00:00 GMT
Location: /colorpictures
Content-Length: 0
Connection: close
Server: AmazonS3
```

Related Resources

- PUT Bucket (p. 144)
- GET Bucket requestPayment (p. 126)

PUT Bucket versioning

Description

This implementation of the PUT operation uses the `versioning` subresource to set the versioning state of an existing bucket. To set the versioning state, you must be the bucket owner.

You can set the versioning state with one of the following values:

- **Enabled—**Enables versioning for the objects in the bucket
  All objects added to the bucket receive a unique version ID.
- **Suspended—**Disables versioning for the objects in the bucket
  All objects added to the bucket receive the version ID `null`.

If the versioning state has never been set on a bucket, it has no versioning state; a GET *`versioning`* request does not return a versioning state value.

If the bucket owner enables MFA Delete in the bucket versioning configuration, the bucket owner must include the *`x-amz-mfa`* request header and the *`Status`* and the *`MfaDelete`* request elements in a request to set the versioning state of the bucket.

For more information about creating a bucket, see PUT Bucket (p. 144). For more information about returning the versioning state of a bucket, see GET Bucket Versioning Status (p. 128).

Requests

Syntax

```
PUT /?versioning HTTP/1.1
Host: BucketName.s3.amazonaws.com
Content-Length: length
Date: date
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))
x-amz-mfa: [SerialNumber] [TokenCode]

<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Status>VersioningState</Status>
  <MfaDelete>MfaDeleteState</MfaDelete>
</VersioningConfiguration>
```

Note the space between *`[SerialNumber]`* and *`[TokenCode]`*.

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

Name	Description	Required
x-amz-mfa	The value is the concatenation of the authentication device's serial number, a space, and the value displayed on your authentication device. Type: String Default: None Condition: Required to configure the versioning state if versioning is configured with MFA Delete enabled.	Conditional

Request Elements

Name	Description	Required
Status	Sets the versioning state of the bucket. Type: Enum Valid Values: Suspended | Enabled Ancestor: VersioningConfiguration	No
MfaDelete	Specifies whether MFA Delete is enabled in the bucket versioning configuration. When enabled, the bucket owner must include the *x-amz-mfa* request header in requests to change the versioning state of a bucket and to permanently delete a versioned object. Type: Enum Valid Values: Disabled | Enabled Ancestor: VersioningConfiguration Constraint: Can only be used when you use *Status*.	No
VersioningConfiguration	Container for setting the versioning state. Type: Container Children: Status Ancestor: None	Yes

Responses

Response Headers

This implementation of the operation uses only response headers that are common to most responses. For more information, see Common Response Headers (p. 14).

Response Elements

This implementation of the operation does not return response elements.

Special Errors

This implementation of the operation does not return special errors. For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

Sample Request

The following request enables versioning for the specified bucket.

```
PUT /?versioning HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 01 Mar  2006 12:00:00 GMT
Authorization: authorization string
Content-Type: text/plain
Content-Length: 124

<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Status>Enabled</Status>
</VersioningConfiguration>
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-id-2: YgIPIfBiKa2bj0KMg95r/0zo3emzU4dzsD4rcKCHQUAdQkf3ShJTOOpXUueF6QKo
x-amz-request-id: 236A8905248E5A01
Date: Wed, 01 Mar  2006 12:00:00 GMT
```

Sample Request

The following request suspends versioning for the specified bucket.

```
PUT /?versioning HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 12 Oct 2009 17:50:00 GMT
Authorization: authorization string
Content-Type: text/plain
Content-Length: 124

<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
   <Status>Suspended</Status>
</VersioningConfiguration>
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-id-2: YgIPIfBiKa2bj0KMg95r/0zo3emzU4dzsD4rcKCHQUAdQkf3ShJTOOpXUueF6QKo
x-amz-request-id: 236A8905248E5A01
Date: Wed, 01 Mar  2006 12:00:00 GMT
```

Sample Request

The following request enables versioning and MFA Delete on a bucket.

```
PUT /?versioning HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 12 Oct 2009 17:50:00 GMT
x-amz-mfa:[SerialNumber] [TokenCode]
Authorization: authorization string
Content-Type: text/plain
Content-Length: 124

<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
   <Status>Enabled</Status>
   <MfaDelete>Enabled</MfaDelete>
</VersioningConfiguration>
```

Note the space between [*SerialNumber*] and [*TokenCode*] and that you must include *Status* whenever you use *MfaDelete*.

Sample Response

```
HTTPS/1.1 200 OK
x-amz-id-2: YgIPIfBiKa2bj0KMg95r/0zo3emzU4dzsD4rcKCHQUAdQkf3ShJTOOpXUueF6QKo
x-amz-request-id: 236A8905248E5A01
Date: Wed, 01 Mar  2006 12:00:00 GMT

Location: /colorpictures
Content-Length: 0
Connection: close
Server: AmazonS3
```

Related Resources

- DELETE Bucket (p. 69)
- PUT Bucket (p. 144)

PUT Bucket website

Description

Sets the configuration of the website that is specified in the *website* subresource. To configure a bucket as a website, you can add this subresource on the bucket with website configuration information such as the file name of the index document and any redirect rules. For more information, go to Hosting Websites on Amazon S3 in the *Amazon Simple Storage Service Developer Guide*.

This PUT operation requires the S3:PutBucketWebsite permission. By default, only the bucket owner can configure the *website* attached to a bucket; however, bucket owners can allow other users to set the *website* configuration by writing a bucket policy that grants them the S3:PutBucketWebsite permission.

Requests

Syntax

```
PUT /?website HTTP/1.1
Host: bucketname.s3.amazonaws.com
Date: date
Content-Length: ContentLength
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))

<WebsiteConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <!-- website configuration information. -->
</WebsiteConfiguration>
```

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

This implementation of the operation uses only request headers that are common to all operations. For more information, see Common Request Headers (p. 12).

Request Elements

You can use a website configuration to redirect all requests to the website endpoint of a bucket, or you can add routing rules that redirect only specific requests.

- To redirect all website requests sent to the bucket's website endpoint, you add a website configuration with the following elements. Because all requests are send to another website, you don't need to provide index document name for the bucket.

Name	Description	Required
WebsiteConfiguration	The root element for the website configuration Type: Container Ancestors: None	Yes

Name	Description	Required
RedirectAllRequestsTo	Describes the redirect behavior for every request to this bucket's website endpoint. If this element is present, no other siblings are allowed. Type: Container Ancestors: WebsiteConfiguration	Yes
HostName	Name of the host where requests will be redirected. Type: String Ancestors: RedirectAllRequestsTo	Yes
Protocol	Protocol to use (http, https) when redirecting requests. The default is the protocol that is used in the original request. Type: String Ancestors: RedirectAllRequestsTo	No

- If you want granular control over redirects, you can use the following elements to add routing rules that describe conditions for redirecting requests and information about the redirect destination. In this case, the website configuration must provide an index document for the bucket, because some requests might not be redirected.

Name	Description	Required
WebsiteConfiguration	Container for the request Type: Container Ancestors: None	Yes
IndexDocument	Container for the *Suffix* element. Type: Container Ancestors: WebsiteConfiguration	Yes
Suffix	A suffix that is appended to a request that is for a *directory* on the website endpoint (e.g., if the suffix is index.html and you make a request to `samplebucket/images/`, the data that is returned will be for the object with the key name images/index.html) The suffix must not be empty and must not include a slash character. Type: String Ancestors: WebsiteConfiguration.IndexDocument	Yes
ErrorDocument	Container for the *Key* element Type: Container Ancestors: WebsiteConfiguration	No

Name	Description	Required
Key	The object key name to use when a 4XX class error occurs. This key identifies the page that is returned when such an error occurs. Type: String Ancestors: WebsiteConfiguration.ErrorDocument Condition: Required when *ErrorDocument* is specified.	Conditional
RoutingRules	Container for a collection of RoutingRule elements. Type: Container Ancestors: WebsiteConfiguration	No
RoutingRule	Container for one routing rule that identifies a condition and a redirect that applies when the condition is met. Type: String Ancestors: WebsiteConfiguration.RoutingRules Condition: In a *RoutingRules* container, there must be at least one of *RoutingRule* element.	Yes
Condition	A container for describing a condition that must be met for the specified redirect to apply. For example: • If request is for pages in the `/docs` folder, redirect to the `/documents` folder. • If request results in HTTP error 4xx, redirect request to another host where you might process the error. Type: Container Ancestors: WebsiteConfiguration.RoutingRules.RoutingRule	No
KeyPrefixEquals	The object key name prefix when the redirect is applied. For example, to redirect requests for `ExamplePage.html`, the key prefix will be `ExamplePage.html`. To redirect request for all pages with the prefix `docs/`, the key prefix will be `/docs`, which identifies all objects in the `docs/` folder. Type: String Ancestors: WebsiteConfiguration.RoutingRules.RoutingRule.Condition Condition: Required when the parent element *Condition* is specified and sibling `HttpErrorCodeReturnedEquals` is not specified. If both conditions are specified, both must be true for the redirect to be applied.	Conditional

Name	Description	Required
HttpErrorCodeReturnedEquals	The HTTP error code when the redirect is applied. In the event of an error, if the error code equals this value, then the specified redirect is applied. Type: String Ancestors: WebsiteConfiguration.RoutingRules.RoutingRule.Condition Condition: Required when parent element *Condition* is specified and sibling KeyPrefixEquals is not specified. If both are specified, then both must be true for the redirect to be applied.	Conditional
Redirect	Container for redirect information. You can redirect requests to another host, to another page, or with another protocol. In the event of an error, you can specify a different error code to return. Type: String Ancestors: WebsiteConfiguration.RoutingRules.RoutingRule	Yes
Protocol	The protocol to use in the redirect request. Type: String Ancestors: WebsiteConfiguration.RoutingRules.RoutingRule.Redirect Valid Values: http, https Condition: Not required if one of the siblings is present	No
HostName	The host name to use in the redirect request. Type: String Ancestors: WebsiteConfiguration.RoutingRules.RoutingRule.Redirect Condition: Not required if one of the siblings is present	No
ReplaceKeyPrefixWith	The object key prefix to use in the redirect request. For example, to redirect requests for all pages with prefix docs/ (objects in the docs/ folder) to documents/, you can set a condition block with *KeyPrefixEquals* set to docs/ and in the Redirect set *ReplaceKeyPrefixWith* to /documents. Type: String Ancestors: WebsiteConfiguration.RoutingRules.RoutingRule.Redirect Condition: Not required if one of the siblings is present. Can be present only if *ReplaceKeyWith* is not provided.	No

Name	Description	Required
ReplaceKeyWith	The specific object key to use in the redirect request. For example, redirect request to `error.html`. Type: String Ancestors: WebsiteConfiguration.RoutingRules.RoutingRule.Redirect Condition: Not required if one of the sibling is present. Can be present only if *ReplaceKeyPrefixWith* is not provided.	No
HttpRedirectCode	The HTTP redirect code to use on the response. Type: String Ancestors: WebsiteConfiguration.RoutingRules.RoutingRule.Redirect Condition: Not required if one of the siblings is present.	No

Responses

Response Headers

This implementation of the operation uses only response headers that are common to most responses. For more information, see Common Response Headers (p. 14).

Response Elements

This implementation of the operation does not return response elements.

Examples

Example 1: Configure bucket as a website (add website configuration)

The following request configures a bucket `example.com` as a website. The configuration in the request specifies index.html as the index document. It also specifies the optional error document, `SomeError-Document.html`.

```
PUT ?website HTTP/1.1
Host: example.com.s3.amazonaws.com
Content-Length: 256
Date: Thu, 27 Jan 2011 12:00:00 GMT
Authorization: signatureValue

<WebsiteConfiguration xmlns='http://s3.amazonaws.com/doc/2006-03-01/'>
    <IndexDocument>
        <Suffix>index.html</Suffix>
    </IndexDocument>
    <ErrorDocument>
        <Key>SomeErrorDocument.html</Key>
    </ErrorDocument>
</WebsiteConfiguration>
```

Amazon S3 returns the following sample response.

```
HTTP/1.1 200 OK
x-amz-id-2: YgIPIfBiKa2bj0KMgUAdQkf3ShJTOOpXUueF6QKo
x-amz-request-id: 80CD4368BD211111
Date: Thu, 27 Jan 2011 00:00:00 GMT
Content-Length: 0
Server: AmazonS3
```

Example 2: Configure bucket as a website but redirect all requests

The following request configures a bucket `www.example.com` as a website; however, the configuration specifies that all GET requests for the `www.example.com` bucket's website endpoint will be redirected to host `example.com`.

```
PUT ?website HTTP/1.1
Host: www.example.com.s3.amazonaws.com
Content-Length: length-value
Date: Thu, 27 Jan 2011 12:00:00 GMT
Authorization: signatureValue

<WebsiteConfiguration xmlns='http://s3.amazonaws.com/doc/2006-03-01/'>
   <RedirectAllRequestsTo>
      <HostName>example.com</HostName>
    </RedirectAllRequestsTo>
</WebsiteConfiguration>
```

This redirect can be useful when you want to serve requests for both `http://www.example.com` and `http://example.com`, but you want to maintain the website content in only one bucket, in this case `example.com`. For more information, go to Hosting Websites on Amazon S3 in the *Amazon Simple Storage Service Developer Guide*.

Example 3: Configure bucket as a website and also specify optional redirection rules

Example 1 is the simplest website configuration. It configures a bucket as a website by providing only an index document and an error document. You can further customize the website configuration by adding routing rules that redirect requests for one or more objects. For example, suppose your bucket contained the following objects:

index.html

docs/article1.html

docs/article2.html

If you decided to rename the folder from `docs/` to `documents/`, you would need to redirect requests for prefix `/docs` to `documents/`. For example, a request for `docs/article1.html` will need to be redirected to `documents/article1.html`.

In this case, you update the website configuration and add a routing rule as shown in the following request:

```
PUT ?website HTTP/1.1
Host: www.example.com.s3.amazonaws.com
Content-Length: length-value
Date: Thu, 27 Jan 2011 12:00:00 GMT
Authorization: signatureValue
```

```
<WebsiteConfiguration xmlns='http://s3.amazonaws.com/doc/2006-03-01/'>
  <IndexDocument>
    <Suffix>index.html</Suffix>
  </IndexDocument>
  <ErrorDocument>
    <Key>Error.html</Key>
  </ErrorDocument>

  <RoutingRules>
    <RoutingRule>
    <Condition>
      <KeyPrefixEquals>docs/</KeyPrefixEquals>
    </Condition>
    <Redirect>
      <ReplaceKeyPrefixWith>documents/</ReplaceKeyPrefixWith>
    </Redirect>
    </RoutingRule>
  </RoutingRules>
</WebsiteConfiguration>
```

Example 4: Configure bucket as a website and redirect errors

You can use a routing rule to specify a condition that checks for a specific HTTP error code. When a page request results in this error, you can optionally reroute requests. For example, you might route requests to another host and optionally process the error. The routing rule in the following requests redirects requests to an EC2 instance in the event of an HTTP error 404. For illustration, the redirect also inserts a object key prefix `report-404/` in the redirect. For example, if you request a page `ExamplePage.html` and it results in a HTTP 404 error, the request is routed to a page `report-404/testPage.html` on the specified EC2 instance. If there is no routing rule and the HTTP error 404 occurred, then `Error.html` would be returned.

```
PUT ?website HTTP/1.1
Host: www.example.com.s3.amazonaws.com
Content-Length: 580
Date: Thu, 27 Jan 2011 12:00:00 GMT
Authorization: signatureValue

<WebsiteConfiguration xmlns='http://s3.amazonaws.com/doc/2006-03-01/'>
  <IndexDocument>
    <Suffix>index.html</Suffix>
  </IndexDocument>
  <ErrorDocument>
    <Key>Error.html</Key>
  </ErrorDocument>

  <RoutingRules>
    <RoutingRule>
    <Condition>
      <HttpErrorCodeReturnedEquals>404</HttpErrorCodeReturnedEquals >
    </Condition>
    <Redirect>
      <HostName>ec2-11-22-333-44.compute-1.amazonaws.com</HostName>
      <ReplaceKeyPrefixWith>report-404/</ReplaceKeyPrefixWith>
    </Redirect>
    </RoutingRule>
```

```
  </RoutingRules>
</WebsiteConfiguration>
```

Example 5: Configure a bucket as a website and redirect folder requests to a page

Suppose you have the following pages in your bucket:

images/photo1.jpg

images/photo2.jpg

images/photo3.jpg

Now you want to route requests for all pages with the `images/` prefix to go to a single page, `errorpage.html`. You can add a website configuration to your bucket with the routing rule shown in the following request:

```
PUT ?website HTTP/1.1
Host: www.example.com.s3.amazonaws.com
Content-Length: 481
Date: Thu, 27 Jan 2011 12:00:00 GMT
Authorization: signatureValue

<WebsiteConfiguration xmlns='http://s3.amazonaws.com/doc/2006-03-01/'>
  <IndexDocument>
    <Suffix>index.html</Suffix>
  </IndexDocument>
  <ErrorDocument>
    <Key>Error.html</Key>
  </ErrorDocument>

  <RoutingRules>
    <RoutingRule>
    <Condition>
      <KeyPrefixEquals>images/</KeyPrefixEquals>
    </Condition>
    <Redirect>
      <ReplaceKeyWith>errorpage.html</ReplaceKeyWith>
    </Redirect>
    </RoutingRule>
  </RoutingRules>
</WebsiteConfiguration>
```

Operations on Objects

Topics

- DELETE Object (p. 200)
- Delete Multiple Objects (p. 203)
- GET Object (p. 212)
- GET Object ACL (p. 222)
- GET Object torrent (p. 226)
- HEAD Object (p. 228)

- OPTIONS object (p. 235)
- POST Object (p. 238)
- POST Object restore (p. 247)
- PUT Object (p. 250)
- PUT Object acl (p. 262)
- PUT Object - Copy (p. 269)
- Initiate Multipart Upload (p. 282)
- Upload Part (p. 290)
- Upload Part - Copy (p. 295)
- Complete Multipart Upload (p. 302)
- Abort Multipart Upload (p. 308)
- List Parts (p. 310)

This section describes operations you can perform on Amazon S3 objects.

DELETE Object

Description

The `DELETE` operation removes the null version (if there is one) of an object and inserts a delete marker, which becomes the current version of the object. If there isn't a null version, Amazon S3 does not remove any objects.

Versioning

To remove a specific version, you must be the bucket owner and you must use the *`versionId`* subresource. Using this subresource permanently deletes the version. If the object deleted is a delete marker, Amazon S3 sets the response header, x-amz-delete-marker, to `true`.

If the object you want to delete is in a bucket where the bucket versioning configuration is MFA Delete enabled, you must include the *`x-amz-mfa`* request header in the `DELETE versionId` request. Requests that include *`x-amz-mfa`* must use HTTPS.

For more information about MFA Delete, go to Using MFA Delete. To see sample requests that use versioning, see Sample Request (p. 202).

You can delete objects by explicitly calling the DELETE Object API or configure its lifecycle (see PUT Bucket lifecycle (p. 162)) to enable Amazon S3 to remove them for you. If you want to block users or accounts from removing or deleting objects from your bucket you must deny them `s3:DeleteObject`, `s3:DeleteObjectVersion` and `s3:PutLifeCycleConfiguration` actions.

Requests

Syntax

```
DELETE /ObjectName HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Content-Length: length
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))
```

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

Name	Description	Required
`x-amz-mfa`	The value is the concatenation of the authentication device's serial number, a space, and the value displayed on your authentication device. Type: String Default: None Condition: Required to permanently delete a versioned object if versioning is configured with MFA Delete enabled.	Conditional

Request Elements

This implementation of the operation does not use request elements.

Responses

Response Headers

Header	Description
x-amz-delete-marker	Specifies whether the versioned object that was permanently deleted was (true) or was not (false) a delete marker. In a simple DELETE, this header indicates whether (true) or not (false) a delete marker was created. Type: Boolean Valid Values: true | false Default: false
x-amz-version-id	Returns the version ID of the delete marker created as a result of the DELETE operation. If you delete a specific object version, the value returned by this header is the version ID of the object version deleted. Type: String Default: None

Response Elements

This implementation of the operation does not return response elements.

Special Errors

This implementation of the operation does not return special errors. For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

Sample Request

The following request deletes the object, my-second-image.jpg.

```
DELETE /my-second-image.jpg HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 12 Oct 2009 17:50:00 GMT
Authorization: authorization string
Content-Type: text/plain
```

Sample Response

```
HTTP/1.1 204 NoContent
x-amz-id-2: LriYPLdmOdAiIfgSm/F1YsViT1LW94/xUQxMsF7xiEb1a0wiIOIxl+zbwZ163pt7
x-amz-request-id: 0A49CE4060975EAC
Date: Wed, 12 Oct 2009 17:50:00 GMT
Content-Length: 0
```

```
Connection: close
Server: AmazonS3
```

Sample Request Deleting a Specified Version of an Object

The following request deletes the specified version of the object, `my-third-image.jpg`.

```
DELETE /my-third-image.jpg?versionId=UIORUnfndfiufdisojhr398493jfdkjFJjkndnqUif
hnw89493jJFJ HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 12 Oct 2009 17:50:00 GMT
Authorization: authorization string
Content-Type: text/plain
Content-Length: 0
```

Sample Response

```
HTTP/1.1 204 NoContent
x-amz-id-2: LriYPLdmOdAiIfgSm/F1YsViT1LW94/xUQxMsF7xiEb1a0wiIOIxl+zbwZ163pt7
x-amz-request-id: 0A49CE4060975EAC
x-amz-version-id: UIORUnfndfiufdisojhr398493jfdkjFJjkndnqUifhnw89493jJFJ
Date: Wed, 12 Oct 2009 17:50:00 GMT
Content-Length: 0
Connection: close
Server: AmazonS3
```

Sample Response if the Object Deleted is a Delete Marker

```
HTTP/1.1 204 NoContent
x-amz-id-2: LriYPLdmOdAiIfgSm/F1YsViT1LW94/xUQxMsF7xiEb1a0wiIOIxl+zbwZ163pt7
x-amz-request-id: 0A49CE4060975EAC
x-amz-version-id: 3/L4kqtJlcpXroDTDmJ+rmSpXd3dIbrHY+MTRCxf3vjVBH40Nr8X8gdRQBpUM
LUo
x-amz-delete-marker: true
Date: Wed, 12 Oct 2009 17:50:00 GMT
Content-Length: 0
Connection: close
Server: AmazonS3
```

Sample Request Deleting a Specified Version of an Object in an MFA-Enabled Bucket

The following request deletes the specified version of the object, `my-third-image.jpg`, which is stored in an MFA-enabled bucket.

```
DELETE /my-third-image.jpg?versionId=UIORUnfndfiuf HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 12 Oct 2009 17:50:00 GMT
x-amz-mfa:[SerialNumber] [AuthenticationCode]
Authorization: authorization string
Content-Type: text/plain
Content-Length: 0
```

Sample Response

```
HTTPS/1.1 204 NoContent
x-amz-id-2: LriYPLdmOdAiIfgSm/F1YsViT1LW94/xUQxMsF7xiEb1a0wiIOIxl+zbwZ163pt7
x-amz-request-id: 0A49CE4060975EAC
x-amz-version-id: UIORUnfndfiuf
Date: Wed, 12 Oct 2009 17:50:00 GMT
Content-Length: 0
Connection: close
Server: AmazonS3
```

Related Resources

- PUT Object (p. 250)
- DELETE Object (p. 200)

Delete Multiple Objects

Description

The Multi-Object Delete operation enables you to delete multiple objects from a bucket using a single HTTP request. If you know the object keys that you want to delete, then this operation provides a suitable alternative to sending individual delete requests (see DELETE Object (p. 200)), reducing per-request overhead.

The Multi-Object Delete request contains a list of up to 1000 keys that you want to delete. In the XML, you provide the object key names, and optionally, version IDs if you want to delete a specific version of the object from a versioning-enabled bucket. For each key, Amazon S3 performs a delete operation and returns the result of that delete, success, or failure, in the response. Note that, if the object specified in the request is not found, Amazon S3 returns the result as deleted.

The Multi-Object Delete operation supports two modes for the response; verbose and quiet. By default, the operation uses verbose mode in which the response includes the result of deletion of each key in your request. In quiet mode the response includes only keys where the delete operation encountered an error. For a successful deletion, the operation does not return any information about the delete in the response body.

When performing a Multi-Object Delete operation on an MFA Delete enabled bucket, that attempts to delete any versioned objects, you must include an MFA token. If you do not provide one, the entire request will fail, even if there are non versioned objects you are attempting to delete. If you provide an invalid token, whether there are versioned keys in the request or not, the entire Multi-Object Delete request will fail. For information about MFA Delete, see MFA Delete.

Finally, the Content-MD5 header is required for all Multi-Object Delete requests. Amazon S3 uses the header value to ensure that your request body has not be altered in transit.

Requests

Syntax

```
POST /?delete HTTP/1.1
Host: bucketname.s3.amazonaws.com
Authorization: authorization string
```

```
Content-Length: Size
Content-MD5: MD5

<?xml version="1.0" encoding="UTF-8"?>
<Delete>
    <Quiet>true</Quiet>
    <Object>
         <Key>Key</Key>
         <VersionId>VersionId</VersionId>
    </Object>
    <Object>
         <Key>Key</Key>
    </Object>
    ...
</Delete>
```

Request Parameters

The Multi-Object Delete operation requires a single query string parameter called "delete" to distinguish it from other bucket POST operations.

Request Headers

This operation uses the following Request Headers in addition to the request headers common to most requests. For more information, see Common Request Headers (p. 12).

Name	Description	Required
Content-MD5	The base64-encoded 128-bit MD5 digest of the data. This header must be used as a message integrity check to verify that the request body was not corrupted in transit. For more information, go to RFC 1864. Type: String Default: None	Yes
Content-Length	Length of the body according to RFC 2616. Type: String Default: None	Yes
x-amz-mfa	The value is the concatenation of the authentication device's serial number, a space, and the value that is displayed on your authentication device. Type: String Default: None Condition: Required to permanently delete a versioned object if versioning is configured with MFA Delete enabled.	Conditional

Request Elements

Name	Description	Required
Delete	Container for the request. Ancestor: None Type: Container Children: One or more *Object* elements and an optional Quiet element.	Yes
Quiet	Element to enable quiet mode for the request. When you add this element, you must set its value to true. Ancestor: *Delete* Type: Boolean Default: false	No
Object	Container element that describes the delete request for an object. Ancestor: *Delete* Type: Container Children: *Key* element and an optional VersionId element.	Yes
Key	Key name of the object to delete. Ancestor: *Object* Type: String	Yes
VersionId	VersionId for the specific version of the object to delete. Ancestor: *Object* Type: String	No

Responses

Response Headers

This operation uses only response headers that are common to most responses. For more information, see Common Response Headers (p. 14).

Response Elements

Name	Description
DeleteResult	Container for the response. Children: *Deleted*, *Error* Type: Container Ancestor: None

Name	Description
`Deleted`	Container element for a successful delete. It identifies the object that was successfully deleted. Children: *`Key`*, *`VersionId`* Type: `Container` Ancestor: *`DeleteResult`*
`Key`	Key name for the object that Amazon S3 attempted to delete. Type: `String` Ancestor: *`Deleted`*, or *`Error`*
`VersionId`	VersionId for the versioned object in the case of a versioned delete. Type: `String` Ancestor: *`Deleted`*
`DeleteMarker`	DeleteMarker element with a true value indicates that the request accessed a delete marker. If a specific delete request either creates or deletes a delete marker, Amazon S3 returns this element in the response with a value of true. This is only the case when your Multi-Object Delete request is on a bucket that has versioning enabled or suspended. For more information about delete markers, go to Object Versioning. Type: `Boolean` Ancestor: *`Deleted`*
`DeleteMarkerVersionId`	Version ID of the delete marker accessed (deleted or created) by the request. If the specific delete request in the Multi-Object Delete either creates or deletes a delete marker, Amazon S3 returns this element in response with the version ID of the delete marker. When deleting an object in a bucket with versioning enabled, this value is present for the following two reasons: • You send a non-versioned delete request, that is, you specify only object key and not the version ID. In this case, Amazon S3 creates a delete marker and returns its version ID in the response. • You send a versioned delete request, that is, you specify an object key and a version ID in your request; however, the version ID identifies a delete marker. In this case, Amazon S3 deletes the delete marker and returns the specific version ID in response. For information about versioning, go to Object Versioning. Type: `String` Ancestor: *`Deleted`*

Name	Description
`Error`	Container for a failed delete operation that describes the object that Amazon S3 attempted to delete and the error it encountered. Children: *`Key`*, *`VersionId`*, `Code`, `Message`. Type: `String` Ancestor: *`DeleteResult`*
`Key`	Key for the object Amazon S3 attempted to delete. Type: `String` Ancestor: `Error`
`VersionId`	Version ID of the versioned object Amazon S3 attempted to delete. Amazon S3 includes this element only in case of a versioned-delete request. Type: String Ancestor: *`Deleted`*, *`Error`*
`Code`	Status code for the result of the failed delete. . Type: *`String`* Values: `AccessDenied`, `InternalError` Ancestor: *`Error`*
`Message`	Error description. Type: `String` Ancestor: *`Error`*

Examples

Example 1: Multi-Object Delete resulting in mixed success/error response

This example illustrates a Multi-Object Delete request to delete objects that result in mixed success and errors response.

Sample Request

The following Multi-Object Delete request deletes two objects from a bucket (bucketname). In this example, the requester does not have permission to delete the sample2.txt object.

```
POST /?delete HTTP/1.1
Host: bucketname.S3.amazonaws.com
Accept: */*
x-amz-date: Wed, 30 Nov 2011 03:39:05 GMT
Content-MD5: p5/WA/oEr30qrEEl21PAqw==
Authorization: AWS AKIAIOSFODNN7EXAMPLE:W0qPYCLe6JwkZAD1ei6hp9XZIee=
Content-Length: 125
Connection: Keep-Alive

<Delete>
  <Object>
    <Key>sample1.txt</Key>
  </Object>
```

```
  <Object>
    <Key>sample2.txt</Key>
  </Object>
</Delete>
```

Sample Response

The response includes a `DeleteResult` element that includes a `Deleted` element for the item that Amazon S3 successfully deleted and an `Error` element that Amazon S3 did not delete because you didn't have permission to delete the object.

```
HTTP/1.1 200 OK
x-amz-id-2: 5h4FxSNCUS7wP5z92eGCWDshNpMnRuXvETa4HH3LvvH6VAIr0jU7tH9kM7X+njXx
x-amz-request-id: A437B3B641629AEE
Date: Fri, 02 Dec 2011 01:53:42 GMT
Content-Type: application/xml
Server: AmazonS3
Content-Length: 251

<?xml version="1.0" encoding="UTF-8"?>
<DeleteResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Deleted>
    <Key>sample1.txt</Key>
  </Deleted>
  <Error>
    <Key>sample2.txt</Key>
    <Code>AccessDenied</Code>
    <Message>Access Denied</Message>
  </Error>
</DeleteResult>
```

Example 2: Deleting Object from a Versioned Bucket

If you delete an item from a versioning enabled bucket, all versions of that object remain in the bucket; however, Amazon S3 inserts a delete marker. For more information, go to Object Versioning.

The following scenarios describe the behavior of a Multi-Object Delete request when versioning is enabled for your bucket.

Case 1 - Simple Delete

The following sample the Multi-Object Delete request specifies only one key.

```
POST /?delete HTTP/1.1
Host: bucketname.S3.amazonaws.com
Accept: */*
x-amz-date: Wed, 30 Nov 2011 03:39:05 GMT
Content-MD5: p5/WA/oEr30qrEEl21PAqw==
Authorization: AWS AKIAIOSFODNN7EXAMPLE:W0qPYCLe6JwkZAD1ei6hp9XZIee=
Content-Length: 79
Connection: Keep-Alive

<Delete>
  <Object>
    <Key>SampleDocument.txt</Key>
```

```
  </Object>
</Delete>
```

Because versioning is enabled on the bucket, Amazon S3 does not delete the object. Instead, it adds a delete marker for this object. The response indicates that a delete marker was added (the `DeleteMarker` element in the response as a value of true) and the version number of the delete marker it added.

```
HTTP/1.1 200 OK
x-amz-id-2: P3xqrhuhYxlrefdw3rEzmJh8z5KDtGzb+/FB7oiQaScI9Yaxd8olYXc7d1111ab+
x-amz-request-id: 264A17BF16E9E80A
Date: Wed, 30 Nov 2011 03:39:32 GMT
Content-Type: application/xml
Server: AmazonS3
Content-Length: 276

<?xml version="1.0" encoding="UTF-8"?>
<DeleteResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Deleted>
    <Key>SampleDocument.txt</Key>
    <DeleteMarker>true</DeleteMarker>
    <DeleteMarkerVersionId>NeQt5xeFTfgPJD8B4CGWnkSLtluMr11s</DeleteMarkerVer
sionId>
  </Deleted>
</DeleteResult>
```

Case 2 - Versioned Delete

The following Multi-Object Delete attempts to delete a specific version of an object

```
POST /?delete HTTP/1.1
Host: bucketname.S3.amazonaws.com
Accept: */*
x-amz-date: Wed, 30 Nov 2011 03:39:05 GMT
Content-MD5: p5/WA/oEr30qrEEl21PAqw==
Authorization: AWS AKIAIOSFODNN7EXAMPLE:W0qPYCLe6JwkZAD1ei6hp9XZIxx=
Content-Length: 140
Connection: Keep-Alive

<Delete>
  <Object>
    <Key>SampleDocument.txt</Key>
    <VersionId>OYcLXagmS.WaD..oyH4KRguB95_YhLs7</VersionId>
  </Object>
</Delete>
```

In this case, Amazon S3 deletes the specific object version from the bucket and returns the following response. In the response, Amazon S3 returns the key and version ID of the object deleted.

```
HTTP/1.1 200 OK
x-amz-id-2: P3xqrhuhYxlrefdw3rEzmJh8z5KDtGzb+/FB7oiQaScI9Yaxd8olYXc7d1111xx+
x-amz-request-id: 264A17BF16E9E80A
Date: Wed, 30 Nov 2011 03:39:32 GMT
Content-Type: application/xml
Server: AmazonS3
Content-Length: 219
```

```
<?xml version="1.0" encoding="UTF-8"?>
<DeleteResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Deleted>
    <Key>SampleDocument.txt</Key>
    <VersionId>OYcLXagmS.WaD..oyH4KRguB95_YhLs7</VersionId>
  </Deleted>
</DeleteResult>
```

Case 3 - Versioned Delete of a Delete Marker

In the preceding example, the request refers to a delete marker (instead of an object), then Amazon S3 deletes the delete marker. The effect of this operation is to make your object reappear in your bucket. Amazon S3 returns a response that indicates the delete marker it deleted (`DeleteMarker` element with value true) and the version ID of the delete marker.

```
HTTP/1.1 200 OK
x-amz-id-2: IIPUZrtolxDEmWsKOae9JlSZe6yWfTye3HQ3T2iAe0ZE4XHa6NKvAJcPp51zZaBr
x-amz-request-id: D6B284CEC9B05E4E
Date: Wed, 30 Nov 2011 03:43:25 GMT
Content-Type: application/xml
Server: AmazonS3
Content-Length: 331

<?xml version="1.0" encoding="UTF-8"?>
<DeleteResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Deleted>
    <Key>SampleDocument.txt</Key>
    <VersionId>NeQt5xeFTfgPJD8B4CGWnkSLtluMr11s</VersionId>
    <DeleteMarker>true</DeleteMarker>
    <DeleteMarkerVersionId>NeQt5xeFTfgPJD8B4CGWnkSLtluMr11s</DeleteMarkerVer
sionId>
  </Deleted>
</DeleteResult>
```

In general, when a Multi-Object Delete request results in Amazon S3 either adding a delete marker or removing a delete marker, the response returns the following elements.

```
<DeleteMarker>true</DeleteMarker>
<DeleteMarkerVersionId>NeQt5xeFTfgPJD8B4CGWnkSLtluMr11s</DeleteMarkerVersionId>
```

Example 3: Malformed XML in the Request

This example shows how Amazon S3 responds to a request that includes a malformed XML document.

Sample Request

The following requests sends a malformed XML document (missing the `Delete` end element).

```
POST /?delete HTTP/1.1
Host: bucketname.S3.amazonaws.com
Accept: */*
x-amz-date: Wed, 30 Nov 2011 03:39:05 GMT
Content-MD5: p5/WA/oEr30qrEEl21PAqw==
```

```
Authorization: AWS AKIAIOSFODNN7EXAMPLE:W0qPYCLe6JwkZAD1ei6hp9XZIee=
Content-Length: 104
Connection: Keep-Alive

<Delete>
  <Object>
    <Key>404.txt</Key>
  </Object>
  <Object>
    <Key>a.txt</Key>
  </Object>
```

Sample Response

The response returns the Error messages that describe the error.

```
HTTP/1.1 200 OK
x-amz-id-2: P3xqrhuhYxlrefdw3rEzmJh8z5KDtGzb+/FB7oiQaScI9Yaxd8olYXc7d1111ab+
x-amz-request-id: 264A17BF16E9E80A
Date: Wed, 30 Nov 2011 03:39:32 GMT
Content-Type: application/xml
Server: AmazonS3
Content-Length: 207

<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>MalformedXML</Code>
  <Message>The XML you provided was not well-formed or did not
           validate against our published schema</Message>
  <RequestId>91F27FB5811111F</RequestId>
  <HostId>LCiQK7KbXyJ1t+tncmjRwmNoeeRNW1/ktJ61IC8kN32SFXJx7UBhOzseJCixAbcD</Host
Id>
</Error>
```

Related Actions

- Initiate Multipart Upload (p. 282)
- Upload Part (p. 290)
- Complete Multipart Upload (p. 302)
- Abort Multipart Upload (p. 308)
- List Parts (p. 310)

GET Object

Description

This implementation of the GET operation retrieves objects from Amazon S3. To use GET, you must have READ access to the object. If you grant READ access to the anonymous user, you can return the object without using an authorization header.

An Amazon S3 bucket has no directory hierarchy such as you would find in a typical computer file system. You can, however, create a logical hierarchy by using object key names that imply a folder structure. For example, instead of naming an object `sample.jpg`, you can name it `photos/2006/February/sample.jpg`.

To get an object from such a logical hierarchy, specify the full key name for the object in the GET operation. For a virtual hosted-style request example, if you have the object `photos/2006/February/sample.jpg`, specify the resource as `/photos/2006/February/sample.jpg`. For a path-style request example, if you have the object `photos/2006/February/sample.jpg` in the bucket named `examplebucket`, specify the resource as `/examplebucket/photos/2006/February/sample.jpg`. For more information about request types, see HTTP Host Header Bucket Specification in the *Amazon Simple Storage Service Developer Guide*.

To distribute large files to many people, you can save bandwidth costs by using BitTorrent. For more information, see Amazon S3 Torrent in the *Amazon Simple Storage Service Developer Guide*. For more information about returning the ACL of an object, see GET Object ACL (p. 222).

If the object you are retrieving is a GLACIER storage class object, the object is archived in Amazon Glacier. You must first restore a copy using the POST Object restore (p. 247) API before you can retrieve the object. Otherwise, this operation returns an `InvalidObjectStateError` error. For information about archiving objects in Amazon Glacier, go to Object Lifecycle Management in the *Amazon Simple Storage Service Developer Guide*.

If you encrypt an object by using server-side encryption with customer-provided encryption keys (SSE-C) when you store the object in Amazon S3, then when you GET the object, you must use the headers documented in the section Specific Request Headers for Server-Side Encryption with Customer-Provided Encryption Keys (p. 215). For more information about SSE-C, go to Server-Side Encryption (Using Customer-Provided Encryption Keys) in the *Amazon Simple Storage Service Developer Guide*.

Permissions

You need the `s3:GetObject` permission for this operation. For more information, go to Specifying Permissions in a Policy in the Amazon Simple Storage Service Developer Guide. If the object you request does not exist, the error Amazon S3 returns depends on whether you also have the `s3:ListBucket` permission.

- If you have the `s3:ListBucket` permission on the bucket, Amazon S3 will return a HTTP status code 404 ("no such key") error.
- if you don't have the `s3:ListBucket` permission, Amazon S3 will return a HTTP status code 403 ("access denied") error.

Versioning

By default, the GET operation returns the current version of an object. To return a different version, use the *versionId* subresource.

Note
If the current version of the object is a delete marker, Amazon S3 behaves as if the object was deleted and includes *x-amz-delete-marker: true* in the response.

For more information about versioning, see PUT Bucket versioning (p. 187) To see sample requests that use versioning, see Sample Request Getting a Specified Version of an Object (p. 219) .

Requests

Syntax

```
GET /ObjectName HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))
Range:bytes=byte_range
```

Request Parameters

There are times when you want to override certain response header values in a GET response. For example, you might override the `Content-Disposition` response header value in your GET request.

You can override values for a set of response headers using the query parameters listed in the following table. These response header values are sent only on a successful request, that is, when status code `200 OK` is returned. The set of headers you can override using these parameters is a subset of the headers that Amazon S3 accepts when you create an object. The response headers that you can override for the GET response are `Content-Type`, `Content-Language`, `Expires`, `Cache-Control`, `Content-Disposition`, and `Content-Encoding`. To override these header values in the GET response, you use the request parameters described in the following table.

Note
You must sign the request, either using an `Authorization` header or a pre-signed URL, when using these parameters. They cannot be used with an unsigned (anonymous) request.

Parameter	Description	Required
response-content-type	Sets the `Content-Type` header of the response. Type: String Default: None	No
response-content-language	Sets the `Content-Language` header of the response. Type: String Default: None	No
response-expires	Sets the `Expires` header of the response. Type: String Default: None	No
response-cache-control	Sets the `Cache-Control` header of the response. Type: String Default: None	No

Parameter	Description	Required
response-content-disposition	Sets the `Content-Disposition` header of the response. Type: String Default: None	No
response-content-encoding	Sets the `Content-Encoding` header of the response. Type: String Default: None	No

Request Headers

This implementation of the operation can use the following request headers in addition to the request headers common to all operations. For more information, see Common Request Headers (p. 12).

Name	Description	Required
Range	Downloads the specified range bytes of an object. For more information about the HTTP Range header, go to http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.35. Type: String Default: None Constraints: None	No
If-Modified-Since	Return the object only if it has been modified since the specified time, otherwise return a 304 (not modified). Type: String Default: None Constraints: None	No
If-Unmodified-Since	Return the object only if it has not been modified since the specified time, otherwise return a 412 (precondition failed). Type: String Default: None Constraints: None	No
If-Match	Return the object only if its entity tag (*ETag*) is the same as the one specified; otherwise, return a 412 (precondition failed). Type: String Default: None Constraints: None	No

Name	Description	Required
If-None-Match	Return the object only if its entity tag (*ETag*) is different from the one specified; otherwise, return a 304 (not modified). Type: String Default: None Constraints: None	No

Specific Request Headers for Server-Side Encryption with Customer-Provided Encryption Keys

When you retrieve an object from Amazon S3 that was encrypted by using server-side encryption with customer-provided encryption keys (SSE-C), you must use the following request headers. For more information about SSE-C, go to Server-Side Encryption (Using Customer-Provided Encryption Keys) in the *Amazon Simple Storage Service Developer Guide*.

Name	Description	Required
x-amz-server-side-encryption-customer-algorithm	Specifies the algorithm to use to when decrypting the requested object. Type: String Default: None Valid Values: AES256 Constraints: Must be accompanied by valid *x-amz-server-side-encryption-customer-key* and *x-amz-server-side-encryption-customer-key-MD5* headers.	Yes
x-amz-server-side-encryption-customer-key	Specifies the customer-provided base64-encoded encryption key to use to decrypt the requested object. This value is used to perform the decryption and then it is discarded; Amazon does not store the key. The key must be appropriate for use with the algorithm specified in the *x-amz-server-side-encryption-customer-algorithm* header. Type: String Default: None Constraints: Must be accompanied by valid *x-amz-server-side-encryption-customer-algorithm* and *x-amz-server-side-encryption-customer-key-MD5* headers.	Yes

Name	Description	Required
x-amz-server-side-encryption-customer-key-MD5	Specifies the base64-encoded 128-bit MD5 digest of the customer provided encryption key according to RFC 1321. Amazon S3 uses this header for a message integrity check to ensure that the encryption key was transmitted without error. Type: String Default: None Constraints: Must be accompanied by valid *x-amz-server-side-encryption-customer-algorithm* and *x-amz-server-side-encryption-customer-key* headers.	Yes

Request Elements

This implementation of the operation does not use request elements.

Responses

Response Headers

Header	Description
x-amz-delete-marker	Specifies whether the object retrieved was (`true`) or was not (`false`) a delete marker. If `false`, this response header does not appear in the response. Type: Boolean Valid Values: `true` | `false` Default: `false`
x-amz-expiration	Amazon S3 returns this header if an `Expiration` action is configured for the object as part of the bucket's lifecycle configuration. The header value includes an "expiry-date" component and a URL-encoded "rule-id" component. Note that for versioning-enabled buckets, this header applies only to current versions; Amazon S3 does not provide a header to infer when a noncurrent version will be eligible for permanent deletion. For more information, see PUT Bucket lifecycle (p. 162). Type: String
x-amz-server-side-encryption	If the object is stored using server-side encryption, the response includes this header with the value of the encryption algorithm used. Type: String Valid Values: AES256
x-amz-server-side-encryption-customer-algorithm	If server-side encryption with customer-provided encryption keys decryption was requested, the response will include this header confirming the decryption algorithm used. Type: String Valid Values: AES256

Header	Description
x-amz-server-side-encryption-customer-key-MD5	If server-side encryption with customer-provided encryption keys decryption was requested, the response includes this header to provide roundtrip message integrity verification of the customer-provided encryption key. Type: String
x-amz-restore	Provides information about the object restoration operation and expiration time of the restored object copy. For more information about archiving objects and restoring them, go to Object Lifecycle Management in *Amazon Simple Storage Service Developer Guide* Type: String Default: None
x-amz-version-id	Returns the version ID of the retrieved object if it has a unique version ID. Type: String Default: None
x-amz-website-redirect-location	When a bucket is configured as a website, you can set this metadata on the object so the website endpoint will evaluate the request for the object as a 301 redirect to another object in the same bucket or an external URL. Type: String Default: None

Response Elements

This implementation of the operation does not return response elements.

Special Errors

This implementation of the operation does not return special errors. For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

Sample Request

The following request returns the object, `my-image.jpg`.

```
GET /my-image.jpg HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: authorization string
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-id-2: eftixk72aD6Ap51TnqcoF8eFidJG9Z/2mkiDFu8yU9AS1ed4OpIszj7UDNEHGran
```

```
x-amz-request-id: 318BC8BC148832E5
Date: Wed, 28 Oct 2009 22:32:00 GMT
Last-Modified: Wed, 12 Oct 2009 17:50:00 GMT
ETag: "fba9dede5f27731c9771645a39863328"
Content-Length: 434234
Content-Type: text/plain
Connection: close
Server: AmazonS3
[434234 bytes of object data]
```

If the object had expiration set using lifecycle configuration, you get the following response with the x-amz-expiration header.

```
HTTP/1.1 200 OK
x-amz-id-2: eftixk72aD6Ap51TnqcoF8eFidJG9Z/2mkiDFu8yU9AS1ed4OpIszj7UDNEHGran
x-amz-request-id: 318BC8BC148832E5
Date: Wed, 28 Oct 2009 22:32:00 GMT
Last-Modified: Wed, 12 Oct 2009 17:50:00 GMT
x-amz-expiration: expiry-date="Fri, 23 Dec 2012 00:00:00 GMT", rule-id="picture-
deletion-rule"
ETag: "fba9dede5f27731c9771645a39863328"
Content-Length: 434234
Content-Type: text/plain
Connection: close
Server: AmazonS3
[434234 bytes of object data]
```

Sample Response if an Object Is Archived in Amazon Glacier

An object archived in Amazon Glacier must first be restored before you can access it. If you attempt to access an Amazon Glacier object without restoring it, Amazon S3 returns the following error.

```
HTTP/1.1 403 Forbidden
x-amz-request-id: CD4BD8A1310A11B3
x-amz-id-2: m9RDbQU0+RRBTjOUN1ChQ1eqMUnr9dv8b+KP6I2gHfRJZSTSrMCoRP8RtPRzX9mb
Content-Type: application/xml
Date: Mon, 12 Nov 2012 23:53:21 GMT
Server: AmazonS3
Content-Length: 231

<Error>
   <Code>InvalidObjectState</Code>
   <Message>The operation is not valid for the object's storage class</Message>

   <RequestId>9FEFFF118E15B86F</RequestId>
   <HostId>WVQ5kzhiT+oiUfDCOiOYv8W4Tk9eNcxWi/MK+hTS/av34Xy4rBU3zsavf0aaaaa</Host
Id>
</Error>
```

Sample Response if the Latest Object Is a Delete Marker

```
HTTP/1.1 404 Not Found
x-amz-request-id: 318BC8BC148832E5
x-amz-id-2: eftixk72aD6Ap51Tnqzj7UDNEHGran
```

```
x-amz-version-id: 3GL4kqtJlcpXroDTDm3vjVBH40Nr8X8g
x-amz-delete-marker:  true
Date: Wed, 28 Oct 2009 22:32:00 GMT
Content-Type: text/plain
Connection: close
Server: AmazonS3
```

Notice that the delete marker returns a 404 Not Found error.

Sample Request Getting a Specified Version of an Object

The following request returns the specified version of an object.

```
GET /myObject?versionId=3/L4kqtJlcpXroDTDmpUMLUo HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: authorization string
```

Sample Response to a Versioned Object GET Request

```
HTTP/1.1 200 OK
x-amz-id-2: eftixk72aD6Ap54OpIszj7UDNEHGran
x-amz-request-id: 318BC8BC148832E5
Date: Wed, 28 Oct 2009 22:32:00 GMT
Last-Modified: Sun, 1 Jan 2006 12:00:00 GMT
x-amz-version-id: 3/L4kqtJlcpXroDTDmJ+rmSpXd3QBpUMLUo
ETag: "fba9dede5f27731c9771645a39863328"
Content-Length: 434234
Content-Type: text/plain
Connection: close
Server: AmazonS3
[434234 bytes of object data]
```

Sample Request with Parameters Altering Response Header Values

The following request specifies all the query string parameters in a GET request overriding the response header values.

```
GET /Junk3.txt?response-cache-control=No-cache&response-content-disposition=at
tachment%3B%20filename%3Dtesting.txt&response-content-encoding=x-gzip&response-
content-language=mi%2C%20en&response-ex
pires=Thu%2C%2001%20Dec%201994%2016:00:00%20GMT HTTP/1.1
x-amz-date: Sun, 19 Dec 2010 01:53:44 GMT
Accept: */*
Authorization: AWS AKIAIOSFODNN7EXAMPLE:aaStE6nKnw8ihhiIdReoXYlMamW=
```

Sample Response with Overridden Response Header Values

In the following sample response note, the header values are set to the values specified in the `true` request.

```
HTTP/1.1 200 OK
x-amz-id-2: SIidWAK3hK+Il3/Qqiu1ZKEuegzLAAspwsgwnwygb9GgFseeFHL5CII8NXSrfWW2
```

```
x-amz-request-id: 881B1CBD9DF17WA1
Date: Sun, 19 Dec 2010 01:54:01 GMT
x-amz-meta-param1: value 1
x-amz-meta-param2: value 2
Cache-Control: No-cache
Content-Language: mi, en
Expires: Thu, 01 Dec 1994 16:00:00 GMT
Content-Disposition: attachment; filename=testing.txt
Content-Encoding: x-gzip
Last-Modified: Fri, 17 Dec 2010 18:10:41 GMT
ETag: "0332bee1a7bf845f176c5c0d1ae7cf07"
Accept-Ranges: bytes
Content-Type: text/plain
Content-Length: 22
Server: AmazonS3

[object data not shown]
```

Sample Request with a Range Header

The following request specifies the HTTP Range header to retrieve the first 10 bytes of an object. For more information about the HTTP Range header, go to http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html.

```
GET /example-object HTTP/1.1
Host: example-bucket.s3.amazonaws.com
x-amz-date: Fri, 28 Jan 2011 21:32:02 GMT
Range: bytes=0-9
Authorization: AWS AKIAIOSFODNN7EXAMPLE:Yxg83MZaEgh3OZ3l0rLo5RTX11o=
Sample Response with Specified Range of the Object Bytes
```

Sample Response

In the following sample response, note that the header values are set to the values specified in the `true` request.

```
HTTP/1.1 206 Partial Content
x-amz-id-2: MzRISOwyjmnupCzjI1WC06l5TTAzm7/JypPGXLh0OVFGcJaaO3KW/hRAqKOpIEEp
x-amz-request-id: 47622117804B3E11
Date: Fri, 28 Jan 2011 21:32:09 GMT
x-amz-meta-title: the title
Last-Modified: Fri, 28 Jan 2011 20:10:32 GMT
ETag: "b2419b1e3fd45d596ee22bdf62aaaa2f"
Accept-Ranges: bytes
Content-Range: bytes 0-9/443
Content-Type: text/plain
Content-Length: 10
Server: AmazonS3

[10 bytes of object data]
```

Sample: Get an Object Stored Using Server-Side Encryption with Customer-Provided Encryption Keys

If an object is stored in Amazon S3 using server-side encryption with customer-provided encryption keys, Amazon S3 needs encryption information so that it can decrypt the object before sending it to you in response to a GET request. You provide the encryption information in your GET request using the relevant headers (see Specific Request Headers for Server-Side Encryption with Customer-Provided Encryption Keys (p. 215)), as shown in the following example request.

```
GET /example-object HTTP/1.1
Host: example-bucket.s3.amazonaws.com

Accept: */*
Authorization:authorization string
Date: Wed, 28 May 2014 19:24:44 +0000
x-amz-server-side-encryption-customer-
key:g0lCfA3Dv40jZz5SQJ1ZukLRFqtI5WorC/8SEKEXAMPLE
x-amz-server-side-encryption-customer-key-MD5:ZjQrne1X/iTcskbY2m3example
x-amz-server-side-encryption-customer-algorithm:AES256
```

The following sample response shows some of the response headers Amazon S3 returns. Note that it includes the encryption information in the response.

```
HTTP/1.1 200 OK
x-amz-id-2: ka5jRm8X3N12ZiY29Z989zg2tNSJPMcK+to7jNjxImXBbyChqc6tLAv+sau7Vjzh
x-amz-request-id: 195157E3E073D3F9
Date: Wed, 28 May 2014 19:24:45 GMT
Last-Modified: Wed, 28 May 2014 19:21:01 GMT
ETag: "c12022c9a3c6d3a28d29d90933a2b096"
x-amz-server-side-encryption-customer-algorithm: AES256
x-amz-server-side-encryption-customer-key-MD5: ZjQrne1X/iTcskbY2m3example
```

Related Resources

- GET Service (p. 65)
- GET Object ACL (p. 222)

GET Object ACL

Description

This implementation of the GET operation uses the *acl* subresource to return the access control list (ACL) of an object. To use this operation, you must have READ_ACP access to the object.

Versioning

By default, GET returns ACL information about the current version of an object. To return ACL information about a different version, use the *versionId* subresource.

To see sample requests that use Versioning, see Sample Request Getting the ACL of the Specific Version of an Object (p. 224).

Requests

Syntax

```
GET /ObjectName?acl HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))
Range:bytes=byte_range
```

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

This implementation of the operation uses only request headers that are common to all operations. For more information, see Common Request Headers (p. 12).

Request Elements

This implementation of the operation does not use request elements.

Responses

Response Headers

This implementation of the operation uses only response headers that are common to most responses. For more information, see Common Response Headers (p. 14).

Response Elements

Name	Description
AccessControlList	Container for Grant, Grantee, and Permission Type: Container Ancestors: AccessControlPolicy

Name	Description
`AccessControlPolicy`	Contains the elements that set the ACL permissions for an object per Grantee. Type: Container Ancestors: None
`DisplayName`	Screen name of the bucket owner Type: String Ancestors: AccessControlPolicy.Owner
`Grant`	Container for the grantee and his or her permissions. Type: Container Ancestors: AccessControlPolicy.AccessControlList
`Grantee`	The subject whose permissions are being set. Type: String Ancestors: AccessControlPolicy.AccessControlList.Grant
`ID`	ID of the bucket owner, or the ID of the grantee Type: String Ancestors: AccessControlPolicy.Owner or AccessControlPolicy.AccessControlList.Grant
`Owner`	Container for the bucket owner's display name and ID. Type: Container Ancestors: AccessControlPolicy
`Permission`	Specifies the permission (FULL_CONTROL, WRITE, READ_ACP) given to the grantee. Type: String Ancestors: AccessControlPolicy.AccessControlList.Grant

Special Errors

This implementation of the operation does not return special errors. For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

Sample Request

The following request returns information, including the ACL, of the object, my-image.jpg.

```
GET /my-image.jpg?acl HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: authorization string
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-id-2: eftixk72aD6Ap51TnqcoF8eFidJG9Z/2mkiDFu8yU9AS1ed4OpIszj7UDNEHGran
x-amz-request-id: 318BC8BC148832E5
x-amz-version-id: 4HL4kqtJlcpXroDTDmJ+rmSpXd3dIbrHY+MTRCxf3vjVBH40Nrjfkd
Date: Wed, 28 Oct 2009 22:32:00 GMT
Last-Modified: Sun, 1 Jan 2006 12:00:00 GMT
Content-Length: 124
Content-Type: text/plain
Connection: close
Server: AmazonS3

<AccessControlPolicy>
  <Owner>
    <ID>75aa57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
    <DisplayName>mtd@amazon.com</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="CanonicalUser">
       <ID>75aa57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>

        <DisplayName>mtd@amazon.com</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

Sample Request Getting the ACL of the Specific Version of an Object

The following request returns information, including the ACL, of the specified version of the object, my-image.jpg.

```
GET /my-image.jpg?versionId=3/L4kqtJlcpXroDVBH40Nr8X8gdRQBpUMLUo&acl HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: authorization string
```

Sample Response Showing the ACL of the Specific Version

```
HTTP/1.1 200 OK
x-amz-id-2: eftixk72aD6Ap51TnqcoF8eFidJG9Z/2mkiDFu8yU9AS1ed4OpIszj7UDNEHGran
x-amz-request-id: 318BC8BC148832E5
Date: Wed, 28 Oct 2009 22:32:00 GMT
Last-Modified: Sun, 1 Jan 2006 12:00:00 GMT
x-amz-version-id: 3/L4kqtJlcpXroDTDmJ+rmSpXd3dIbrHY+MTRCxf3vjVBH40Nr8X8gdRQBpUM
LUo
Content-Length: 124
Content-Type: text/plain
Connection: close
Server: AmazonS3
```

```
<AccessControlPolicy>
  <Owner>
    <ID>75aa57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
    <DisplayName>mdtd@amazon.com</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="CanonicalUser">
       <ID>75aa57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>

        <DisplayName>mdtd@amazon.com</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

Related Resources

- GET Object (p. 212)
- PUT Object (p. 250)
- DELETE Object (p. 200)

GET Object torrent

Description

This implementation of the GET operation uses the *torrent* subresource to return torrent files from a bucket. BitTorrent can save you bandwidth when you're distributing large files. For more information about BitTorrent, see Amazon S3 Torrent.

> **Note**
> You can get torrent only for objects that are less than 5 GB in size and that are not encrypted using server-side encryption with customer-provided encryption key.

To use GET, you must have READ access to the object.

Requests

Syntax

```
GET /ObjectName?torrent HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))
```

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

This implementation of the operation uses only request headers that are common to all operations. For more information, see Common Request Headers (p. 12).

Request Elements

This implementation of the operation does not use request elements.

Responses

Response Headers

This implementation of the operation uses only response headers that are common to most responses. For more information, see Common Response Headers (p. 14).

Response Elements

This implementation of the operation does not return response elements.

Special Errors

This implementation of the operation does not return special errors. For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

Getting Torrent Files in a Bucket

This example retrieves the Torrent file for the "Nelson" object in the "quotes" bucket.

```
GET /quotes/Nelson?torrent HTTP/1.0
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: authorization string
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-request-id: 7CD745EBB7AB5ED9
Date: Wed, 25 Nov 2009 12:00:00 GMT
Content-Disposition: attachment; filename=Nelson.torrent;
Content-Type: application/x-bittorrent
Content-Length: 537
Server: AmazonS3

<body: a Bencoded dictionary as defined by the BitTorrent specification>
```

Related Resources

- GET Object (p. 212)

HEAD Object

Description

The HEAD operation retrieves metadata from an object without returning the object itself. This operation is useful if you are interested only in an object's metadata. To use HEAD, you must have READ access to the object.

A HEAD request has the same options as a GET operation on an object. The response is identical to the GET response except that there is no response body.

If you encrypt an object by using server-side encryption with customer-provided encryption keys (SSE-C) when you store the object in Amazon S3, then when you retrieve the metadata from the object, you must use the headers documented in the section Specific Request Headers for Server-Side Encryption with Customer-Provided Encryption Keys (p. 229). For more information about SSE-C, go to Server-Side Encryption (Using Customer-Provided Encryption Keys) in the *Amazon Simple Storage Service Developer Guide*.

Permissions

You need the `s3:GetObject` permission for this operation. For more information, go to Specifying Permissions in a Policy in the Amazon Simple Storage Service Developer Guide. If the object you request does not exist, the error Amazon S3 returns depends on whether you also have the `s3:ListBucket` permission.

- If you have the `s3:ListBucket` permission on the bucket, Amazon S3 will return a HTTP status code 404 ("no such key") error.
- if you don't have the `s3:ListBucket` permission, Amazon S3 will return a HTTP status code 403 ("access denied") error.

Versioning

By default, the HEAD operation retrieves metadata from the current version of an object. If the current version is a delete marker, Amazon S3 behaves as if the object was deleted. To retrieve metadata from a different version, use the *versionId* subresource. For more information, see Versions in the *Amazon Simple Storage Service Developer Guide*.

To see sample requests that use versioning, see Sample Request Getting Metadata from a Specified Version of an Object (p. 233).

Requests

Syntax

```
HEAD /ObjectName HTTP/1.1
Host: BucketName.s3.amazonaws.com
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))
Date: date
```

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

This implementation of the operation can use the following request headers in addition to the request headers common to all operations. For more information, see Common Request Headers (p. 12).

Name	Description	Required
Range	Downloads the specified range bytes of an object. For more information about the HTTP Range header, go to http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.35. Type: String Default: None Constraints: None	No
If-Modified-Since	Return the object only if it has been modified since the specified time, otherwise return a 304 (not modified). Type: String Default: None Constraints: None	No
If-Unmodified-Since	Return the object only if it has not been modified since the specified time, otherwise return a 412 (precondition failed). Type: String Default: None Constraints: None	No
If-Match	Return the object only if its entity tag (*ETag*) is the same as the one specified; otherwise, return a 412 (precondition failed). Type: String Default: None Constraints: None	No
If-None-Match	Return the object only if its entity tag (*ETag*) is different from the one specified; otherwise, return a 304 (not modified). Type: String Default: None Constraints: None	No

Specific Request Headers for Server-Side Encryption with Customer-Provided Encryption Keys

When you retrieve metadata from an object stored in Amazon S3 that was encrypted by using server-side encryption with customer-provided encryption keys (SSE-C), you must use the following request headers. For more information about SSE-C, go to Server-Side Encryption (Using Customer-Provided Encryption Keys) in the *Amazon Simple Storage Service Developer Guide*.

Name	Description	Required
x-amz-server-side-encryption-customer-algorithm	Specifies the algorithm to use to when decrypting the requested object. Type: String Default: None Valid Values: AES256 Constraints: Must be accompanied by valid *x-amz-server-side-encryption-customer-key* and *x-amz-server-side-encryption-customer-key-MD5* headers.	Yes
x-amz-server-side-encryption-customer-key	Specifies the customer-provided base64-encoded encryption key to use to decrypt the requested object. This value is used to perform the decryption and then it is discarded; Amazon does not store the key. The key must be appropriate for use with the algorithm specified in the *x-amz-server-side-encryption-customer-algorithm* header. Type: String Default: None Constraints: Must be accompanied by valid *x-amz-server-side-encryption-customer-algorithm* and *x-amz-server-side-encryption-customer-key-MD5* headers.	Yes
x-amz-server-side-encryption-customer-key-MD5	Specifies the base64-encoded 128-bit MD5 digest of the customer provided encryption key according to RFC 1321. Amazon S3 uses this header for a message integrity check to ensure that the encryption key was transmitted without error. Type: String Default: None Constraints: Must be accompanied by valid *x-amz-server-side-encryption-customer-algorithm* and *x-amz-server-side-encryption-customer-key* headers.	Yes

Request Elements

This implementation of the operation does not use request elements.

Responses

Response Headers

This implementation of the operation can include the following response headers in addition to the response headers common to all responses. For more information, see Common Response Headers (p. 14).

Name	Description
x-amz-expiration	Amazon S3 will return this header if an `Expiration` action is configured for the object as part of the bucket's lifecycle configuration. The header value includes an "expiry-date" component and a URL-encoded "rule-id" component. Note that for versioning-enabled buckets, this header applies only to current versions; Amazon S3 does not provide a header to infer when a noncurrent version will be eligible for permanent deletion. For more information, see PUT Bucket lifecycle (p. 162). Type: String
*x-amz-meta-**	If you supplied user metadata in a `PUT` object operation, that metadata is returned in one or more response headers prefixed with `x-amz-meta-` and with the suffix name that you provided on storage. For example, for family, the response header would be `x-amz-meta-family`. Amazon S3 returns this metadata verbatim; Amazon S3 does not interpret it. Type: String
x-amz-missing-meta	This header is set to the number of metadata entries that were not returned in `x-amz-meta` headers. This can happen if you create metadata using an API like SOAP that supports more flexible metadata than the REST API. For example, with SOAP, you can create metadata with values that are not valid HTTP headers. Type: String
x-amz-restore	If the object is an archived object (an object whose storage class is `GLACIER`), the response includes this header if either the archive restoration is in progress (see POST Object restore (p. 247)) or an archive copy is already restored. If an archive copy is already restored, the header value indicates when Amazon S3 is scheduled to delete the object copy. For example, `x-amz-restore: ongoing-request="false", expiry-date="Fri, 23 Dec 2012 00:00:00 GMT"` If the object restoration is in progress, the header will return the value `ongoing-request="true"`. For more information about archiving objects, go to Object Lifecycle Management in *Amazon Simple Storage Service Developer Guide* Type: String Default: None
x-amz-server-side-encryption	If the object is stored by using server-side encryption, the response includes this header with the value of the encryption algorithm that was used. Type: String Valid Values: `AES256`
x-amz-server-side-encryption-customer-algorithm	If server-side encryption with customer-provided encryption keys(SSE-C) decryption was requested, the response will include this header confirming the decryption algorithm used. Type: String Valid Values: `AES256`

Name	Description
x-amz-server-side-encryption-customer-key-MD5	If SSE-C decryption was requested, the response includes this header to provide roundtrip message integrity verification of the customer-provided encryption key. Type: String
x-amz-version-id	The version ID of the object returned. Type: String

Response Elements

Response Elements

This implementation of the operation does not return response elements.

Special Errors

This implementation of the operation does not return special errors. For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

Sample Request

The following request returns the metadata of an object.

```
HEAD /my-image.jpg HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:02236Q3V0RonhpaBX5sCYVf1bNRuU=
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-id-2: ef8yU9AS1ed4OpIszj7UDNEHGran
x-amz-request-id: 318BC8BC143432E5
x-amz-version-id: 3HL4kqtJlcpXroDTDmjVBH40Nrjfkd
Date: Wed, 28 Oct 2009 22:32:00 GMT
Last-Modified: Sun, 1 Jan 2006 12:00:00 GMT
ETag: "fba9dede5f27731c9771645a39863328"
Content-Length: 434234
Content-Type: text/plain
Connection: close
Server: AmazonS3
```

If the object is scheduled to expire according to a lifecycle configuration set on the bucket, the response returns the `x-amz-expiration` tag with information about when Amazon S3 will delete the object. For more information, go to Object Expiration in the *Amazon Simple Storage Service Developer Guide*.

```
HTTP/1.1 200 OK
x-amz-id-2: azQRZtQJ2m1P8R+TIsG9h0VuC/DmiSJmjXUMq7snk+LKSJeurtmfzSlGhR46GzSJ
x-amz-request-id: 0EFF61CCE3F24A26
```

```
Date: Mon, 17 Dec 2012 02:26:39 GMT
Last-Modified: Mon, 17 Dec 2012 02:14:10 GMT
x-amz-expiration: expiry-date="Fri, 21 Dec 2012 00:00:00 GMT", rule-id="Rule
for testfile.txt"
ETag: "54b0c58c7ce9f2a8b551351102ee0938"
Accept-Ranges: bytes
Content-Type: text/plain
Content-Length: 14
Server: AmazonS3
```

Sample Request Getting Metadata from a Specified Version of an Object

The following request returns the metadata of the specified version of an object.

```
HEAD /my-image.jpg?versionId=3HL4kqCxf3vjVBH40Nrjfkd HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:02236Q3V0WpaBX5sCYVf1bNRuU=
```

Sample Response to a Versioned HEAD Request

```
HTTP/1.1 200 OK
x-amz-id-2: eftixk72aD6Ap51TnqcoF8epIszj7UDNEHGran
x-amz-request-id: 318BC8BC143432E5
x-amz-version-id: 3HL4kqtJlcpXrof3vjVBH40Nrjfkd
Date: Wed, 28 Oct 2009 22:32:00 GMT
Last-Modified: Sun, 1 Jan 2006 12:00:00 GMT
ETag: "fba9dede5f27731c9771645a39863328"
Content-Length: 434234
Content-Type: text/plain
Connection: close
Server: AmazonS3
```

Sample Request for an Amazon Glacier Object

For an archived object, the `x-amz-restore` header provides the date when the restored copy expires, as shown in the following response. Even if the object is stored in Amazon Glacier, all object metadata is still available.

```
HEAD /my-image.jpg HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: 13 Nov 2012 00:28:38 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:02236Q3V0RonhpaBX5sCYVf1bNRuU=
```

Sample Response - Glacier Object

If the object is already restored, the `x-amz-restore` header provides the date when the restored copy will expire, as shown in the following response.

```
HTTP/1.1 200 OK
x-amz-id-2: FSVaTMjrmBp3Izs1NnwBZeu7M19iI8UbxMbi0A8AirHANJBo+hEftBuiESACOMJp
x-amz-request-id: E5CEFCB143EB505A
```

```
Date: Tue, 13 Nov 2012 00:28:38 GMT
Last-Modified: Mon, 15 Oct 2012 21:58:07 GMT
x-amz-restore: ongoing-request="false", expiry-date="Wed, 07 Nov 2012 00:00:00
 GMT"
ETag: "1accb31fcf202eba0c0f41fa2f09b4d7"
Accept-Ranges: bytes
Content-Type: binary/octet-stream
Content-Length: 300
Server: AmazonS3
```

If the restoration is in progress, then the `x-amz-restore` header returns a message accordingly.

```
HTTP/1.1 200 OK
x-amz-id-2: b+V2mDiMHTdy1myoUBpctvmJl95H9U/OSUm/jRtHxjh0+pCk5SvByL4xu2TDv4GM
x-amz-request-id: E2E7B6AEE4E9BD2B
Date: Tue, 13 Nov 2012 00:43:32 GMT
Last-Modified: Sat, 20 Oct 2012 21:28:27 GMT
x-amz-restore: ongoing-request="true"
ETag: "1accb31fcf202eba0c0f41fa2f09b4d7"
Accept-Ranges: bytes
Content-Type: binary/octet-stream
Content-Length: 300
Server: AmazonS3
```

Related Resources

- GET Object (p. 212)

OPTIONS object

Description

A browser can send this preflight request to Amazon S3 to determine if it can send an actual request with the specific origin, HTTP method, and headers.

Amazon S3 supports cross-origin resource sharing (CORS) by enabling you to add a `cors` subresource on a bucket. When a browser sends this preflight request, Amazon S3 responds by evaluating the rules that are defined in the `cors` configuration.

If `cors` is not enabled on the bucket, then Amazon S3 returns a `403 Forbidden` response.

For more information about CORS, go to Enabling Cross-Origin Resource Sharing in the *Amazon Simple Storage Service Developer Guide*.

Requests

Syntax

```
OPTIONS /ObjectName HTTP/1.1
Host: BucketName.s3.amazonaws.com
Origin: Origin
Access-Control-Request-Method: HTTPMethod
Access-Control-Request-Headers: RequestHeader
```

Request Parameters

This operation does not introduce any specific request parameters, but it may contain any request parameters that are required by the actual request.

Request Headers

Name	Description	Required
Origin	Identifies the origin of the cross-origin request to Amazon S3. For example, http://www.example.com. Type: String Default: None	Yes
Access-Control-Request-Method	Identifies what HTTP method will be used in the actual request. Type: String Default: None	Yes

Name	Description	Required
Access-Control-Request-Headers	A comma-delimited list of HTTP headers that will be sent in the actual request. For example, to put an object with server-side encryption, this preflight request will determine if it can include the `x-amz-server-side-encryption` header with the request. Type: String Default: None	No

Request Elements

This implementation of the operation does not use request elements.

Responses

Response Headers

Header	Description
`Access-Control-Allow-Origin`	The origin you sent in your request. If the origin in your request is not allowed, Amazon S3 will not include this header in the response. Type: String
`Access-Control-Max-Age`	How long, in seconds, the results of the preflight request can be cached. Type: String
`Access-Control-Allow-Methods`	The HTTP method that was sent in the original request. If the method in the request is not allowed, Amazon S3 will not include this header in the response. Type: String
`Access-Control-Allow-Headers`	A comma-delimited list of HTTP headers that the browser can send in the actual request. If any of the requested headers is not allowed, Amazon S3 will not include that header in the response, nor will the response contain any of the headers with the `Access-Control` prefix. Type: String
`Access-Control-Expose-Headers`	A comma-delimited list of HTTP headers. This header provides the JavaScript client with access to these headers in the response to the actual request. Type: String

Response Elements

This implementation of the operation does not return response elements.

Examples

Example : Send a preflight OPTIONS request to a `cors` enabled bucket

A browser can send this preflight request to Amazon S3 to determine if it can send the actual PUT request from http://www.example.com origin to the Amazon S3 bucket named `examplebucket`.

Sample Request

```
OPTIONS /exampleobject HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Origin: http://www.example.com
Access-Control-Request-Method: PUT
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-id-2: 6SvaESv3VULYPLik5LLl7lSPPtSnBvDdGmnklX1HfUl7uS2m1DF6td6KWKNjYMXZ
x-amz-request-id: BDC4B83DF5096BBE
Date: Wed, 21 Aug 2012 23:09:55 GMT
Etag: "1f1a1af1f1111111111111c11aed1da1"
Access-Control-Allow-Origin: http://www.example.com
Access-Control-Allow-Methods: PUT
Access-Control-Expose-Headers: x-amz-request-id
Content-Length: 0
Server: AmazonS3
```

Related Resources

- GET Bucket cors (p. 92)
- DELETE Bucket cors (p. 71)
- PUT Bucket cors (p. 157)

POST Object

Description

The POST operation adds an object to a specified bucket using HTML forms. POST is an alternate form of PUT that enables browser-based uploads as a way of putting objects in buckets. Parameters that are passed to PUT via HTTP Headers are instead passed as form fields to POST in the multipart/form-data encoded message body. You must have WRITE access on a bucket to add an object to it. Amazon S3 never stores partial objects: if you receive a successful response, you can be confident the entire object was stored.

Amazon S3 is a distributed system. If Amazon S3 receives multiple write requests for the same object simultaneously, all but the last object written will be overwritten.

To ensure that data is not corrupted traversing the network, use the Content-MD5 form field. When you use this form field, Amazon S3 checks the object against the provided MD5 value. If they do not match, Amazon S3 returns an error. Additionally, you can calculate the MD5 value while posting an object to Amazon S3 and compare the returned *ETag* to the calculated MD5 value. The ETag only reflects changes to the contents of an object, not its metadata.

> **Note**
> To configure your application to send the Request Headers prior to sending the request body, use the 100-continue HTTP status code. For POST operations, this helps you avoid sending the message body if the message is rejected based on the headers (e.g., authentication failure or redirect). For more information on the 100-continue HTTP status code, go to Section 8.2.3 of http://www.ietf.org/rfc/rfc2616.txt.

You can optionally request server-side encryption where Amazon S3 encrypts your data as it writes it to disks in its data centers and decrypts it for you when you access it. You have option of providing your own encryption key or you can use the AWS-managed encryption keys. For more information, go to Using Server-Side Encryption in the *Amazon Simple Storage Service Developer Guide*.

Versioning

If you enable versioning for a bucket, POST automatically generates a unique version ID for the object being added. Amazon S3 returns this ID in the response using the *x-amz-version-id* response header.

If you suspend versioning for a bucket, Amazon S3 always uses null as the version ID of the object stored in a bucket.

For more information about returning the versioning state of a bucket, see GET Bucket (Versioning Status) (p. 128).

Amazon S3 is a distributed system. If you enable versioning for a bucket and Amazon S3 receives multiple write requests for the same object simultaneously, all of the objects will be stored.

To see sample requests that use versioning, see Sample Request (p. 245).

Requests

Syntax

```
POST / HTTP/1.1
Host: destinationBucket.s3.amazonaws.com
User-Agent: browser_data
```

```
Accept: file_types
Accept-Language: Regions
Accept-Encoding: encoding
Accept-Charset: character_set
Keep-Alive: 300
Connection: keep-alive
Content-Type: multipart/form-data; boundary=9431149156168
Content-Length: length

--9431149156168
Content-Disposition: form-data; name="key"



acl
--9431149156168
Content-Disposition: form-data; name="success_action_redirect"

success_redirect
--9431149156168
Content-Disposition: form-data; name="Content-Type"

content_type
--9431149156168
Content-Disposition: form-data; name="x-amz-meta-uuid"

uuid
--9431149156168
Content-Disposition: form-data; name="x-amz-meta-tag"

metadata
--9431149156168
Content-Disposition: form-data; name="AWSAccessKeyId"

access-key-id
--9431149156168
Content-Disposition: form-data; name="Policy"

encoded_policy
--9431149156168
Content-Disposition: form-data; name="Signature"

signature=
--9431149156168
Content-Disposition: form-data; name="file"; filename="MyFilename.jpg"
Content-Type: image/jpeg

file_content
--9431149156168
Content-Disposition: form-data; name="submit"

Upload to Amazon S3
--9431149156168--
```

Request Parameters

This implementation of the operation does not use request parameters.

Form Fields

This operation can use the following form fields.

Name	Description	Required
AWSAccessKeyId	The AWS access key ID of the owner of the bucket who grants an Anonymous user access for a request that satisfies the set of constraints in the policy. Type: String Default: None Constraints: Required if a policy document is included with the request.	Conditional
acl	Specifies an Amazon S3 access control list. If an invalid access control list is specified, an error is generated. For more information on ACLs, go to Access Control List (ACL) Overview in the *Amazon Simple Storage Service Developer Guide*. Type: String Default: private Valid Values: `private | public-read | public-read-write | authenticated-read | bucket-owner-read | bucket-owner-full-control`	No
Cache-Control, *Content-Type*, *Content-Disposition*, *Content-Encoding*, *Expires*	REST-specific headers. For more information, see PUT Object (p. 250). Type: String Default: None	No
file	File or text content. The file or text content must be the last field in the form. You cannot upload more than one file at a time. Type: File or text content Default: None	Yes
key	The name of the uploaded key. To use the file name provided by the user, use the ${filename} variable. For example, if the user Betty uploads the file `lolcatz.jpg` and you specify `/user/betty/${filename}`, the key name will be `/user/betty/lolcatz.jpg`. For more information, go to Object Key and Metadata in the *Amazon Simple Storage Service Developer Guide*. Type: String Default: None	Yes

Name	Description	Required
policy	Security Policy describing what is permitted in the request. Requests without a security policy are considered anonymous and only work on publicly writable buckets. For more information, go to HTML Forms and Upload Examples. Type: String Default: None Constraints: Policy is required if the bucket is not publicly writable.	Conditional
success_action_redirect, redirect	The URL to which the client is redirected upon successful upload. If success_action_redirect is not specified, Amazon S3 returns the empty document type specified in the success_action_status field. If Amazon S3 cannot interpret the URL, it acts as if the field is not present. If the upload fails, Amazon S3 displays an error and does not redirect the user to a URL. Type: String Default: None **Note** The redirect field name is deprecated and support for the redirect field name will be removed in the future.	No
success_action_status	The status code returned to the client upon successful upload if success_action_redirect is not specified. Accepts the values 200, 201, or 204 (default). If the value is set to 200 or 204, Amazon S3 returns an empty document with a 200 or 204 status code. If the value is set to 201, Amazon S3 returns an XML document with a 201 status code. If the value is not set or if it is set to an invalid value, Amazon S3 returns an empty document with a 204 status code. Type: String Default: None **Note** Some versions of the Adobe Flash player do not properly handle HTTP responses with an empty body. To support uploads through Adobe Flash, we recommend setting *success_action_status* to 201.	No

Name	Description	Required
x-amz-storage-class	Storage class to use for storing the object. Type: String Default: STANDARD Valid Values: STANDARD | REDUCED_REDUNDANCY Constraints: You cannot specify `GLACIER` as the storage class. To transition objects to the GLACIER storage class you can use lifecycle configuration.	No
*x-amz-meta-**	Field names prefixed with *x-amz-meta-* contain user-specified metadata. Amazon S3 does not validate or use this data. For more information, see PUT Object (p. 250). Type: String Default: None	No
x-amz-security-token	Amazon DevPay security token. Each request that uses Amazon DevPay requires two `x-amz-security-token` form fields: one for the product token and one for the user token. For more information, go to Using DevPay. Type: String Default: None	No

x-amz-website-redirect-location	If the bucket is configured as a website, redirects requests for this object to another object in the same bucket or to an external URL. Amazon S3 stores the value of this header in the object metadata. For information about object metadata, go to Object Key and Metadata.
	
	In the following example, the request header sets the redirect to an object (`anotherPage.html`) in the same bucket:
	`x-amz-website-redirect-location: /anotherPage.html`
	
	In the following example, the request header sets the object redirect to another website:
	
	`x-amz-website-redirect-location: http://www.example.com/`
	
	For more information about website hosting in Amazon S3, go to sections Hosting Websites on Amazon S3 and How to Configure Website Page Redirects in the *Amazon Simple Storage Service Developer Guide*.
	Type: String
	Default: None
	Constraints: The value must be prefixed by, "/", "http://" or "https://". The length of the value is limited to 2 K.
		No

Server-Side Encryption Specific Request Form Fields

You can optionally request Amazon S3 to encrypt data at rest using server-side encryption. Server-side encryption is about data encryption at rest, that is, Amazon S3 encrypts your data as it writes it to disks in its data centers and decrypts it for you when you access it. Depending on whether you want to use AWS-managed encryption keys or provide your own encryption keys, you use the following form fields:

- Use AWS-managed encryption keys — If you want Amazon S3 to manage keys used to encrypt data, you specify the following form fields in the request.

Name	Description	Required
x-amz-server-side-encryption	Specifies a server-side encryption algorithm to use when Amazon S3 creates an object. Type: String Valid Value: AES256	Yes

- Use customer-provided encryption keys — If you want to manage your own encryption keys, you must provide all the following form fields in the request.

  Note
  If you use this feature, the ETag value that Amazon S3 returns in the response will not be the MD5 of the object.

Name	Description	Required
x-amz-server-side-encryption-customer-algorithm	Specifies the algorithm to use to when encrypting the object. Type: String Default: None Valid Value: AES256 Constraints: Must be accompanied by valid *x-amz-server-side-encryption-customer-key* and *x-amz-server-side-encryption-customer-key-MD5* fields.	Yes
x-amz-server-side-encryption-customer-key	Specifies the customer provided base-64 encoded encryption key for Amazon S3 to use in encrypting data. This value is used to store the object and then it is discarded; Amazon does not store the encryption key. The key must be appropriate for use with the algorithm specified in the *x-amz-server-side-encryption-customer-algorithm* header. Type: String Default: None Constraints: Must be accompanied by valid *x-amz-server-side-encryption-customer-algorithm* and *x-amz-server-side-encryption-customer-key-MD5* fields.	Yes

Name	Description	Required
x-amz-server-side-encryption-customer-key-MD5	Specifies the base64-encoded 128-bit MD5 digest of the encryption key according to RFC 1321. Amazon S3 uses this header for a message integrity check to ensure the encryption key was transmitted without error. Type: String Default: None Constraints: Must be accompanied by valid *x-amz-server-side-encryption-customer-algorithm* and *x-amz-server-side-encryption-customer-key* fields.	Yes

Responses

Response Headers

This implementation of the operation can include the following response headers in addition to the response headers common to all responses. For more information, see Common Response Headers (p. 14).

Name	Description
x-amz-expiration	Amazon S3 will return this header if an `Expiration` action is configured for the object as part of the bucket's lifecycle configuration. The header value includes an "expiry-date" component and a URL encoded "rule-id" component. Note that for version-enabled buckets, this header only applies to current versions; Amazon S3 does not provide a header to infer when a noncurrent version will be eligible for permanent deletion. For more information, see PUT Bucket lifecycle (p. 162). Type: String
success_action_redirect, redirect	The URL to which the client is redirected on successful upload. Type: String Ancestor: PostResponse
x-amz-server-side-encryption	If you request server-side encryption when adding an object, the response includes this header confirming the encryption algorithm used. Type: String
x-amz-server-side-encryption-customer-algorithm	If server-side encryption with customer-provided encryption keys (SSE-C) encryption was requested, the response will include this header confirming the encryption algorithm used. Type: String Valid Values: `AES256`

Name	Description
x-amz-server-side-encryption -customer-key-MD5	If SSE-C encryption was requested, the response includes this header to provide round trip message integrity verification of the customer provided encryption key. Type: String
x-amz-version-id	Version of the object. Type: String

Response Elements

Name	Description
Bucket	Name of the bucket the object was stored in. Type: String Ancestor: PostResponse
ETag	The entity tag is an MD5 hash of the object that you can use to do conditional GET operations using the If-Modified request tag with the GET request operation. The ETag reflects changes to only the contents of an object, not its metadata. Type: String Ancestor: PostResponse
Key	The object key name. Type: String Ancestor: PostResponse
Location	URI of the object. Type: String Ancestor: PostResponse

Special Errors

This implementation of the operation does not return special errors. For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

Sample Request

```
POST /Neo HTTP/1.1
Content-Length: 4
Host: quotes.s3.amazonaws.com
Date: Wed, 01 Mar  2006 12:00:00 GMT
Authorization: authorization string
Content-Type: text/plain
Expect: the 100-continue HTTP status code

ObjectContent
```

Sample Response with Versioning Suspended

The following shows a sample response when bucket versioning is suspended.

```
HTTP/1.1 100 Continue
HTTP/1.1 200 OK
x-amz-id-2: LriYPLdmOdAiIfgSm/F1YsViT1LW94/xUQxMsF7xiEb1a0wiIOIxl+zbwZ163pt7
x-amz-request-id: 0A49CE4060975EAC
x-amz-version-id: default
Date: Wed, 12 Oct 2009 17:50:00 GMT
ETag: "1b2cf535f27731c974343645a3985328"
Content-Length: 0
Connection: close
Server: AmazonS3
```

Notice in this response the version ID is `null`.

Sample Response with Versioning Enabled

The following shows a sample response when bucket versioning is enabled.

```
HTTP/1.1 100 Continue
HTTP/1.1 200 OK
x-amz-id-2: LriYPLdmOdAiIfgSm/F1YsViT1LW94/xUQxMsF7xiEb1a0wiIOIxl+zbwZ163pt7
x-amz-request-id: 0A49CE4060975EAC
x-amz-version-id: 43jfkodU8493jnFJD9fjj3HHNVfdsQUIFDNsidf038jfdsjGFDSIRp
Date: Wed, 01 Mar  2006 12:00:00 GMT
ETag: "828ef3fdfa96f00ad9f27c383fc9ac7f"
Content-Length: 0
Connection: close
Server: AmazonS3
```

Related Resources

- PUT Object - Copy (p. 269)
- POST Object (p. 238)
- GET Object (p. 212)

POST Object restore

Description

Restores a temporary copy of an archived object. You can optionally provide version ID to restore specific object version. If version ID is not provided, it will restore the current version.

In the request, you specify the number of days that you want the restored copy to exist. After the specified period, Amazon S3 deletes the temporary copy. Note that the object remains archived; Amazon S3 deletes only the restored copy.

An object in the Glacier storage class is an archived object. To access the object, you must first initiate a restore request, which restores a copy of the archived object. Restore jobs typically complete in three to five hours.

For more information about archiving objects, go to Object Lifecycle Management in *Amazon Simple Storage Service Developer Guide*.

You can obtain restoration status by sending a HEAD request. In the response, these operations return the `x-amz-restore` header with restoration status information.

After restoring an object copy, you can update the restoration period by reissuing this request with the new period. Amazon S3 updates the restoration period relative to the current time and charges only for the request, and there are no data transfer charges.

You cannot issue another restore request when Amazon S3 is actively processing your first restore request for the same object; however, after Amazon S3 restores a copy of the object, you can send restore requests to update the expiration period of the restored object copy.

If your bucket has a lifecycle configuration with a rule that includes an expiration action, the object expiration overrides the life span that you specify in a restore request. For example, if you restore an object copy for 10 days but the object is scheduled to expire in 3 days, Amazon S3 deletes the object in 3 days. For more information about lifecycle configuration, see PUT Bucket lifecycle (p. 162).

To use this action, you must have `s3:RestoreObject` permissions on the specified object. For more information, go to Access Control section in the *Amazon S3 Developer Guide*.

Requests

Syntax

```
POST /ObjectName?restore&versionId=VersionID HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))
Content-MD5: MD5

<RestoreRequest xmlns="http://s3.amazonaws.com/doc/2006-3-01">
   <Days>NumberOfDays</Days>
</RestoreRequest>
```

Note
The syntax shows some of the request headers. For a complete list, see the Request Headers section.

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

Name	Description	Required
Content-MD5	The base64-encoded 128-bit MD5 digest of the data. This header must be used as a message integrity check to verify that the request body was not corrupted in transit. For more information, go to RFC 1864. Type: String Default: None	Yes

Request Elements

Name	Description
RestoreRequest	Container for restore information Type: Container Ancestors: `AccessControlPolicy`
Days	Lifetime of the restored (active) copy. The minimum number of days that you can restore an object from Amazon Glacier is 1. After the object copy reaches the specified lifetime, Amazon S3 removes the copy from the bucket. Type: Positive integer Ancestors: RestoreRequest

Responses

A successful operation returns either `200 OK` or `202 Accepted` status code.

- If the object copy is not previously restored, then Amazon S3 returns `202 Accepted` in the response.
- If the object copy is previously restored, Amazon S3 returns `200 OK` in the response.

Response Headers

This implementation of the operation uses only response headers that are common to most responses. For more information, see Common Response Headers (p. 14).

Response Elements

This operation does not return response elements.

Special Errors

Error Code	Description	HTTP Status Code	SOAP Fault Code Prefix
`RestoreAlreadyInProgress`	Object restore is already in progress.	409 Conflict	Client

Examples

Restore an object for 2 days

The following restore request restores a copy of the `photo1.jpg` object from Amazon Glacier for a period of 2 days.

```
POST /photo1.jpg?restore HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Mon, 22 Oct 2012 01:49:52 GMT
Authorization: authorization string
Content-Length: 53

<RestoreRequest>
  <Days>2</Days>
</RestoreRequest>
```

If the `examplebucket` does not have a restored copy of the object, Amazon S3 returns the following `202 Accepted` response.

```
HTTP/1.1 202 Accepted
x-amz-id-2: GFi
hv3y6+kE7KG11GEkQhU7/2/cHR3Yb2fCb2S04nxI423Dqwg2XiQ0B/UZlzYQvPiBlZNRcovw=
x-amz-request-id: 9F341CD3C4BA79E0
Date: Sat, 20 Oct 2012 23:54:05 GMT
Content-Length: 0
Server: AmazonS3
```

If a copy of the object is already restored, Amazon S3 returns a `200 OK` response, only updating the restored copy's expiry time.

Related Resources

- GET Bucket lifecycle (p. 95)
- PUT Bucket lifecycle (p. 162)

PUT Object

Description

This implementation of the PUT operation adds an object to a bucket. You must have WRITE permissions on a bucket to add an object to it.

Amazon S3 never adds partial objects; if you receive a success response, Amazon S3 added the entire object to the bucket.

Amazon S3 is a distributed system. If it receives multiple write requests for the same object simultaneously, it overwrites all but the last object written. Amazon S3 does not provide object locking; if you need this, make sure to build it into your application layer or use versioning instead.

To ensure that data is not corrupted traversing the network, use the Content-MD5 header. When you use this header, Amazon S3 checks the object against the provided MD5 value and, if they do not match, returns an error. Additionally, you can calculate the MD5 while putting an object to Amazon S3 and compare the returned ETag to the calculated MD5 value.

> **Note**
> To configure your application to send the Request Headers prior to sending the request body, use the 100-continue HTTP status code. For PUT operations, this helps you avoid sending the message body if the message is rejected based on the headers (e.g., because of authentication failure or redirect). For more information on the 100-continue HTTP status code, go to Section 8.2.3 of http://www.ietf.org/rfc/rfc2616.txt.

You can optionally request server-side encryption where Amazon S3 encrypts your data as it writes it to disks in its data centers and decrypts it for you when you access it. You have option to provide your own encryption key or use AWS-managed encryption keys. For more information, go to Using Server-Side Encryption in the *Amazon Simple Storage Service Developer Guide*.

Versioning

If you enable versioning for a bucket, Amazon S3 automatically generates a unique version ID for the object being stored. Amazon S3 returns this ID in the response using the *x-amz-version-id* response header. If versioning is suspended, Amazon S3 always uses null as the version ID for the object stored. For more information about returning the versioning state of a bucket, see GET Bucket versioning (p. 128).

If you enable versioning for a bucket, when Amazon S3 receives multiple write requests for the same object simultaneously, it stores all of the objects.

To see sample requests that use versioning, see Sample Request (p. 259).

Reduced Redundancy Storage

Reduced redundancy storage (RRS) enables customers to reduce their costs by storing noncritical, reproducible data at lower levels of redundancy than Amazon S3's standard storage. RRS provides a cost-effective, highly available solution for distributing or sharing content that is durably stored elsewhere, or for storing thumbnails, transcoded media, or other processed data that can be easily reproduced. The RRS option stores objects on multiple devices across multiple facilities, providing 400 times the durability of a typical disk drive, but does not replicate objects as many times as standard Amazon S3 storage. Thus, using RRS is even more cost effective.

To store an object using reduced redundancy, set the *x-amz-storage-class* request header to REDUCED_REDUNDANCY. The default value is STANDARD.

Access Permissions

When uploading an object, you can optionally specify the accounts or groups that should be granted specific permissions on your object. There are two ways to grant the appropriate permissions using the request headers:

- Specify a canned (predefined) ACL using the `x-amz-acl` request header. For more information, see Canned ACL in the *Amazon Simple Storage Service Developer Guide*.
- Specify access permissions explicitly using the `x-amz-grant-read`, , `x-amz-grant-read-acp`, and `x-amz-grant-write-acp`, `x-amz-grant-full-control` headers. These headers map to the set of permissions Amazon S3 supports in an ACL. For more information, go to Access Control List (ACL) Overview in the *Amazon Simple Storage Service Developer Guide*.

Note
You can either use a canned ACL or specify access permissions explicitly. You cannot do both.

Requests

Syntax

```
PUT /ObjectName HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))
```

Note
The syntax shows some of the request headers. For a complete list, see the Request Headers section.

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

This implementation of the operation can use the following request headers in addition to the request headers common to all operations. For more information, see Common Request Headers (p. 12).

Name	Description	Required
Cache-Control	Can be used to specify caching behavior along the request/reply chain. For more information, go to http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.9. Type: String Default: None Constraints: None	No

Name	Description	Required
Content-Disposition	Specifies presentational information for the object. For more information, go to http://www.w3.org/Protocols/rfc2616/rfc2616-sec19.html#sec19.5.1. Type: String Default: None Constraints: None	No
Content-Encoding	Specifies what content encodings have been applied to the object and thus what decoding mechanisms must be applied to obtain the media-type referenced by the *Content-Type* header field. For more information, go to http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.11. Type: String Default: None Constraints: None	No
Content-Length	The size of the object, in bytes. For more information, go to http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.13. Type: String Default: None Constraints: None	Yes
Content-MD5	The base64-encoded 128-bit MD5 digest of the message (without the headers) according to RFC 1864. This header can be used as a message integrity check to verify that the data is the same data that was originally sent. Although it is optional, we recommend using the Content-MD5 mechanism as an end-to-end integrity check. For more information about REST request authentication, go to REST Authentication in the Amazon Simple Storage Service Developer Guide Type: String Default: None Constraints: None	No
Content-Type	A standard MIME type describing the format of the contents. For more information, go to http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.17. Type: String Default: binary/octet-stream Valid Values: MIME types Constraints: None	No
Expect	When your application uses `100-continue`, it does not send the request body until it receives an acknowledgment. If the message is rejected based on the headers, the body of the message is not sent. Type: String Default: None Valid Values: `100-continue` Constraints: None	No

Name	Description	Required
Expires	The date and time at which the object is no longer cacheable. For more information, go to http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.21. Type: String Default: None Constraints: None	No
x-amz-meta-	Any header starting with this prefix is considered user metadata. It will be stored with the object and returned when you retrieve the object. The PUT request header is limited to 8 KB in size. Within the PUT request header, the user-defined metadata is limited to 2 KB in size. User-defined metadata is a set of key-value pairs. The size of user-defined metadata is measured by taking the sum of the number of bytes in the UTF-8 encoding of each key and value. Type: String Default: None Constraints: None	No
x-amz-storage-class	RRS enables customers to reduce their costs by storing noncritical, reproducible data at lower levels of redundancy than Amazon S3's standard storage. Type: Enum Default: `STANDARD` Valid Values: `STANDARD` | `REDUCED_REDUNDANCY` Constraints: You cannot specify `GLACIER` as the storage class. To transition objects to the `GLACIER` storage class you can use lifecycle configuration.	No

Name	Description	Required

x-amz-website-redirect-location	If the bucket is configured as a website, redirects requests for this object to another object in the same bucket or to an external URL. Amazon S3 stores the value of this header in the object metadata. For information about object metadata, go to Object Key and Metadata.
	
	In the following example, the request header sets the redirect to an object (`anotherPage.html`) in the same bucket:
	`x-amz-website-redirect-location: /anotherPage.html`
	
	In the following example, the request header sets the object redirect to another website:
	
	`x-amz-website-redirect-location: http://www.example.com/`
	
	For more information about website hosting in Amazon S3, go to sections Hosting Websites on Amazon S3 and How to Configure Website Page Redirects in the *Amazon Simple Storage Service Developer Guide*.
	Type: String
	Default: None
	Constraints: The value must be prefixed by, "/", "http://" or "https://". The length of the value is limited to 2 KB.
		No

Access Control List (ACL) Specific Request Headers

Additionally, you can use the following access control related headers with this operation. By default, all objects are private: only the owner has full control. When adding a new object, you can grant permissions to individual AWS accounts or predefined Amazon S3 groups. These permissions are then used to create the Access Control List (ACL) on the object. For more information, go to Using ACLs.

You can use one of the following two ways to grant these permissions:

- **Specify a canned ACL —** Amazon S3 supports a set of predefined ACLs, known as canned ACLs. Each canned ACL has a predefined set of grantees and permissions. For more information, go to Canned ACL.

Name	Description	Required
x-amz-acl	The canned ACL to apply to the object. For more information, see Canned ACL in the *Amazon Simple Storage Service Developer Guide*. Type: String Default: private Valid Values: `private | public-read | public-read-write | authenticated-read | bucket-owner-read | bucket-owner-full-control` Constraints: None	No

- **Specify access permissions explicitly —** If you want to explicitly grant access permissions to specific AWS accounts or a group, you use the following headers. Each of the following headers maps to specific permissions Amazon S3 supports in an ACL. For more information, go to Access Control List (ACL) Overview. In the header value, you specify a list of grantees who get the specific permission.

Name	Description	Required
x-amz-grant-read	Allows grantee to read the object data and its metadata. Type: String Default: None Constraints: None	No
x-amz-grant-write	Not applicable. This applies only when granting permission on a bucket. Type: String Default: None Constraints: None	No
x-amz-grant-read-acp	Allows grantee to read the object ACL. Type: String Default: None Constraints: None	No
x-amz-grant-write-acp	Allows grantee to write the ACL for the applicable object. Type: String Default: None Constraints: None	No
x-amz-grant-full-control	Allows grantee the READ, READ_ACP, and WRITE_ACP permissions on the object. Type: String Default: None Constraints: None	No

You specify each grantee as a `type=value` pair, where the type can be one of the following:

- **emailAddress** – if value specified is the email address of an AWS account
- **id** – if value specified is the canonical user ID of an AWS account
- **uri** – if granting permission to a predefined group.

For example, the following `x-amz-grant-read` header grants read object data and its metadata permission to the AWS accounts identified by their email addresses.

```
x-amz-grant-read: emailAddress="xyz@amazon.com", emailAddress="abc@amazon.com"
```

Server-Side Encryption Specific Request Headers

You can optionally request Amazon S3 to encrypt data at rest using server-side encryption. Server-side encryption is about data encryption at rest, that is, Amazon S3 encrypts your data as it writes it to disks

in its data centers and decrypts it for you when you access it. Depending on whether you want to use AWS-managed encryption keys or provide your own encryption keys, you use the following headers:

- Use AWS-managed encryption keys — If you want Amazon S3 to manage keys used to encrypt data, you specify the following header in the request.

Name	Description	Required
x-amz-server-side-encryption	Specifies a server-side encryption algorithm to use when Amazon S3 creates an object. Type: String Valid Value: AES256	Yes

- Use customer-provided encryption keys— If you want to manage your own encryption keys, you must provide all the following headers in the request.

  Note
  If you use this feature, the ETag value that Amazon S3 returns in the response will not be the MD5 of the object.

Name	Description	Required
x-amz-server-side-encryption-customer-algorithm	Specifies the algorithm to use to when encrypting the object. Type: String Default: None Valid Value: AES256 Constraints: Must be accompanied by valid *x-amz-server-side-encryption-customer-key* and *x-amz-server-side-encryption-customer-key-MD5* headers.	Yes
x-amz-server-side-encryption-customer-key	Specifies the customer provided base-64 encoded encryption key for Amazon S3 to use in encrypting data. This value is used to store the object and then is discarded; Amazon does not store the encryption key. The key must be appropriate for use with the algorithm specified in the *x-amz-server-side-encryption-customer-algorithm* header. Type: String Default: None Constraints: Must be accompanied by valid *x-amz-server-side-encryption-customer-algorithm* and *x-amz-server-side-encryption-customer-key-MD5* headers.	Yes

Name	Description	Required
x-amz-server-side-encryption-customer-key-MD5	Specifies the base64-encoded 128-bit MD5 digest of the encryption key according to RFC 1321. Amazon S3 uses this header for a message integrity check to ensure the encryption key was transmitted without error. Type: String Default: None Constraints: Must be accompanied by valid *x-amz-server-side-encryption-customer-algorithm* and *x-amz-server-side-encryption-customer-key* headers.	Yes

Responses

Response Headers

This implementation of the operation can include the following response headers in addition to the response headers common to all responses. For more information, see Common Response Headers (p. 14).

Name	Description
x-amz-expiration	If the object expiration is configured (see PUT Bucket lifecycle (p. 162)), the response includes this header. It includes the `expiry-date` and `rule-id` key-value pairs providing object expiration information. The value of the `rule-id` is URL encoded. Type: String
x-amz-server-side-encryption	If you request server-side encryption when adding an object, the response includes this header confirming the encryption algorithm used. Type: String
x-amz-server-side-encryption-customer-algorithm	If server-side encryption with customer-provided encryption keys encryption was requested the response will include this header confirming the encryption algorithm used. Type: String Valid Values: `AES256`
x-amz-server-side-encryption-customer-key-MD5	If server-side encryption using customer provided encryption key was requested the response returns this header to provide round trip message integrity verification of the customer provided encryption key. Type: String
`x-amz-version-id`	Version of the object. Type: String

Response Elements

This implementation of the operation does not return response elements.

Special Errors

This implementation of the operation does not return special errors. For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

Sample Request

The following request stores the image `my-image.jpg` in the bucket `myBucket`.

```
PUT /my-image.jpg HTTP/1.1
Host: myBucket.s3.amazonaws.com
Date: Wed, 12 Oct 2009 17:50:00 GMT
Authorization: authorization string
Content-Type: text/plain
Content-Length: 11434
Expect: 100-continue
[11434 bytes of object data]
```

Sample Response with Versioning Suspended

```
HTTP/1.1 100 Continue

HTTP/1.1 200 OK
x-amz-id-2: LriYPLdmOdAiIfgSm/F1YsViT1LW94/xUQxMsF7xiEb1a0wiIOIxl+zbwZ163pt7
x-amz-request-id: 0A49CE4060975EAC
Date: Wed, 12 Oct 2009 17:50:00 GMT
ETag: "1b2cf535f27731c974343645a3985328"
Content-Length: 0
Connection: close
Server: AmazonS3
```

If an expiration rule created on the bucket using lifecycle configuration applies to the object, you get a response with an `x-amz-expiration` header as shown in the following response. For more information, go to Object Expiration.

```
HTTP/1.1 100 Continue

HTTP/1.1 200 OK
x-amz-id-2: LriYPLdmOdAiIfgSm/F1YsViT1LW94/xUQxMsF7xiEb1a0wiIOIxl+zbwZ163pt7
x-amz-request-id: 0A49CE4060975EAC
Date: Wed, 12 Oct 2009 17:50:00 GMT
x-amz-expiration: expiry-date="Fri, 23 Dec 2012 00:00:00 GMT", rule-id="1"
ETag: "1b2cf535f27731c974343645a3985328"
Content-Length: 0
Connection: close
Server: AmazonS3
```

Sample Response with Versioning Enabled

If the bucket has versioning enabled, the response includes the `x-amz-version-id` header.

```
HTTP/1.1 100 Continue

HTTP/1.1 200 OK
x-amz-id-2: LriYPLdmOdAiIfgSm/F1YsViT1LW94/xUQxMsF7xiEb1a0wiIOIxl+zbwZ163pt7
x-amz-request-id: 0A49CE4060975EAC
x-amz-version-id: 43jfkodU8493jnFJD9fjj3HHNVfdsQUIFDNsidf038jfdsjGFDSIRp
Date: Wed, 12 Oct 2009 17:50:00 GMT
ETag: "fbacf535f27731c9771645a39863328"
Content-Length: 0
Connection: close
Server: AmazonS3
```

Sample Request: Specifying reduced redundancy storage class

The following request stores the image, `my-image.jpg`, in the bucket, `myBucket`. The request specifies `x-amz-storage-class` header to request object be stored using the `REDUCED_REDUNDANCY` storage class.

```
PUT /my-image.jpg HTTP/1.1
Host: myBucket.s3.amazonaws.com
Date: Wed, 12 Oct 2009 17:50:00 GMT
Authorization: authorization string
Content-Type: image/jpeg
Content-Length: 11434
Expect: 100-continue
x-amz-storage-class: REDUCED_REDUNDANCY
```

Sample Response

```
HTTP/1.1 100 Continue

HTTP/1.1 200 OK
x-amz-id-2: LriYPLdmOdAiIfgSm/F1YsViT1LW94/xUQxMsF7xiEb1a0wiIOIxl+zbwZ163pt7
x-amz-request-id: 0A49CE4060975EAC
Date: Wed, 12 Oct 2009 17:50:00 GMT
ETag: "1b2cf535f27731c974343645a3985328"
Content-Length: 0
Connection: close
Server: AmazonS3
```

Sample Request: Uploading an object and specifying access permissions explicitly

The following request stores the file `TestObject.txt` in the bucket `myBucket`. The request specifies various ACL headers to grant permission to AWS accounts specified using canonical user ID and email address.

```
PUT TestObject.txt HTTP/1.1
Host: myBucket.s3.amazonaws.com
x-amz-date: Fri, 13 Apr 2012 05:40:14 GMT
Authorization: authorization string
x-amz-grant-write-acp: id=8a6925ce4adf588a4532142d3f74dd8c71fa124ExampleCanon
icalUserID
x-amz-grant-full-control: emailAddress="ExampleUser@amazon.com"
```

```
x-amz-grant-write: emailAddress="ExampleUser1@amazon.com", emailAddress="Ex
ampleUser2@amazon.com"
Content-Length: 300
Expect: 100-continue
Connection: Keep-Alive

...Object data in the body...
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-id-2: RUxG2sZJUfS+ezeAS2i0Xj6w/ST6xqF/8pFNHjTjTrECW56SCAUWGg+7QLVoj1GH
x-amz-request-id: 8D017A90827290BA
Date: Fri, 13 Apr 2012 05:40:25 GMT
ETag: "dd038b344cf9553547f8b395a814b274"
Content-Length: 0
Server: AmazonS3
```

Sample Request: Using a canned ACL to set access permissions

The following request stores the file `TestObject.txt` in the bucket `myBucket`. The request uses an `x-amz-acl` header to specify a canned ACL to grant READ permission to the public.

```
...Object data in the body...
PUT TestObject.txt HTTP/1.1
Host: myBucket.s3.amazonaws.com
x-amz-date: Fri, 13 Apr 2012 05:54:57 GMT
x-amz-acl: public-read
Authorization: authorization string
Content-Length: 300
Expect: 100-continue
Connection: Keep-Alive

...Object data in the body...
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-id-2: Yd6PSJxJFQeTYJ/3dDO7miqJfVMXXW0S2Hijo3WFs4bz6oe2QCVXasxXLZdMfASd
x-amz-request-id: 80DF413BB3D28A25
Date: Fri, 13 Apr 2012 05:54:59 GMT
ETag: "dd038b344cf9553547f8b395a814b274"
Content-Length: 0
Server: AmazonS3
```

Sample: Upload an object, and request server-side encryption with a customer-provided encryption key

In this upload object example, you request server-side encryption and provide an encryption key.

```
PUT /example-object HTTP/1.1
Host: example-bucket.s3.amazonaws.com
Accept: */*
```

```
Authorization:authorization string
Date: Wed, 28 May 2014 19:31:11 +0000
x-amz-server-side-encryption-customer-
key:g0lCfA3Dv40jZz5SQJ1ZukLRFqtI5WorC/8SEEXAMPLE
x-amz-server-side-encryption-customer-key-MD5:ZjQrne1X/iTcskbY2example
x-amz-server-side-encryption-customer-algorithm:AES256
```

In the response, Amazon S3 returns the encryption algorithm and MD5 of the encryption key you specified when uploading the object. Note that the ETag returned is not the MD5 of the object.

```
HTTP/1.1 200 OK
x-amz-id-2: 7qoYGN7uMuFuYS6m7a4lszH6in+hccE+4DXPmDZ7C9KqucjnZC1gI5mshai6fbMG

x-amz-request-id: 06437EDD40C407C7
Date: Wed, 28 May 2014 19:31:12 GMT
x-amz-server-side-encryption-customer-algorithm: AES256
x-amz-server-side-encryption-customer-key-MD5: ZjQrne1X/iTcskbY2example
ETag: "ae89237c20e759c5f479ece02c642f59"
```

Related Resources

- PUT Object - Copy (p. 269)
- POST Object (p. 238)
- GET Object (p. 212)

PUT Object acl

Description

This implementation of the PUT operation uses the *acl* subresource to set the access control list (ACL) permissions for an object that already exists in a bucket. You must have WRITE_ACP permission to set the ACL of an object.

You can use one of the following two ways to set an object's permissions:

- Specify the ACL in the request body, or
- Specify permissions using request headers

Depending on your application needs, you may choose to set the ACL on an object using either the request body or the headers. For example, if you have an existing application that updates an object ACL using the request body, then you can continue to use that approach.

Versioning

The ACL of an object is set at the object version level. By default, PUT sets the ACL of the current version of an object. To set the ACL of a different version, use the *versionId* subresource.

To see sample requests that use versioning, see Sample Request: Setting the ACL of a specified object version (p. 267).

Requests

Syntax

The following request shows the syntax for sending the ACL in the request body. If you want to use headers to specify the permissions for the object, you cannot send the ACL in the request body. Instead, see the Request Headers section for a list of headers you can use.

```
PUT /ObjectName?acl HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))

<AccessControlPolicy>
  <Owner>
    <ID>ID</ID>
    <DisplayName>EmailAddress</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="CanonicalUser">
        <ID>ID</ID>
        <DisplayName>EmailAddress</DisplayName>
      </Grantee>
      <Permission>Permission</Permission>
    </Grant>
    ...
```

```
  </AccessControlList>
</AccessControlPolicy>
```

Note
The syntax shows some of the request headers. For a complete list see the Request Headers section.

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

You can use the following request headers in addition to the Common Request Headers (p. 12).

These headers enable you to set access permissions using one of the following methods:

- Specify canned ACL, or
- Specify the permission for each grantee explicitly

Amazon S3 supports a set of predefined ACLs, known as canned ACLs. Each canned ACL has a pre-defined a set of grantees and permissions. For more information, see Canned ACL. To grant access permissions by specifying canned ACLs, you use the following header and specify the canned ACL name as its value. If you use this header, you cannot use other access control-specific headers in your request.

Name	Description	Required
x-amz-acl	The canned ACL to apply to the object. For more information, go to Canned ACL in the Amazon Simple . Type: String Valid Values: `private | public-read | public-read-write | authenticated-read | bucket-owner-read | bucket-owner-full-control` Default: private	No

If you need to grant individualized access permissions on an object, you can use the following `x-amz-grant-`*`permission`* headers. When using these headers you specify explicit access permissions and grantees (AWS accounts or Amazon S3 groups) who will receive the permission. If you use these ACL specific headers, you cannot use *x-amz-acl* header to set a canned ACL.

Note
Each of the following request headers maps to specific permissions Amazon S3 supports in an ACL. For more information, go to Access Control List (ACL) Overview.

Name	Description	Required
x-amz-grant-read	Allows the specified grantee to list the objects in the bucket. Type: String Default: None Constraints: None	No

Name	Description	Required
x-amz-grant-write	Not applicable when granting access permissions on objects. You can use this when granting access permissions on buckets. Type: String Default: None Constraints: None	No
x-amz-grant-read-acp	Allows the specified grantee to read the bucket ACL. Type: String Default: None Constraints: None	No
x-amz-grant-write-acp	Allows the specified grantee to write the ACL for the applicable bucket. Type: String Default: None Constraints: None	No
x-amz-grant-full-control	Allows the specified grantee the READ, WRITE, READ_ACP, and WRITE_ACP permissions on the bucket. Type: String Default: None Constraints: None	No

For each of these headers, the value is a comma-separated list of one or more grantees. You specify each grantee as a `type=value` pair, where the type can be one of the following:

- **emailAddress —** if value specified is the email address of an AWS account
- **id —** if value specified is the canonical user ID of an AWS account
- **uri —** if granting permission to a predefined group.

For example, the following `x-amz-grant-read` header grants list objects permission to the two AWS accounts identified by their email addresses.

```
x-amz-grant-read: emailAddress="xyz@amazon.com", emailAddress="abc@amazon.com"
```

For more information, go to Access Control List (ACL) Overview.

Request Elements

If you decide to use the request body to specify an ACL, you must use the following elements.

Note
If you use the request body, you cannot use the request headers to set an ACL.

Name	Description
AccessControlList	Container for ACL information Type: Container Ancestors: `AccessControlPolicy`

Name	Description
AccessControlPolicy	Contains the elements that set the ACL permissions for an object per grantee Type: Container Ancestors: None
DisplayName	Screen name of the bucket owner Type: String Ancestors: `AccessControlPolicy.Owner`
Grant	Container for the grantee and his or her permissions Type: Container Ancestors: `AccessControlPolicy.AccessControlList`
Grantee	The subject whose permissions are being set. Type: String Valid Values: `DisplayName` | `EmailAddress` | `AuthenticatedUser`. For more information, see Grantee Values (p. 265). Ancestors: AccessControlPolicy.AccessControlList.Grant
ID	ID of the bucket owner, or the ID of the grantee Type: String Ancestors: `AccessControlPolicy.Owner` or `AccessControlPolicy.AccessControlList.Grant`
Owner	Container for the bucket owner's display name and ID Type: Container Ancestors: `AccessControlPolicy`
Permission	Specifies the permission given to the grantee Type: String Valid Values: FULL_CONTROL | WRITE | WRITE_ACP | READ | READ_ACP Ancestors: `AccessControlPolicy.AccessControlList.Grant`

Grantee Values

You can specify the person (grantee) to whom you're assigning access rights (using request elements) in the following ways:

- By the person's ID:

```
<Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="Ca
nonicalUser"><ID>ID</ID><DisplayName>GranteesEmail</DisplayName>
</Grantee>
```

  DisplayName is optional and ignored in the request.

- By Email address:

```
<Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="AmazonCustomerByEmail"><EmailAddress>Grantees@email.com</EmailAd
dress>lt;/Grantee>
```

The grantee is resolved to the *CanonicalUser* and, in a response to a `GET Object acl` request, appears as the *CanonicalUser*.

- By URI:

```
<Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="Group"><URI>http://acs.amazonaws.com/groups/global/AuthenticatedUsers</URI></Grantee>
```

Responses

Response Headers

This implementation of the operation can include the following response headers in addition to the response headers common to all responses. For more information, see Common Response Headers (p. 14).

Name	Description
x-amz-version-id	Version of the object whose ACL is being set. Type: String Default: None

Response Elements

This operation does not return response elements.

Special Errors

This operation does not return special errors. For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

Sample Request

The following request grants access permission to an existing object. The request specifies the ACL in the body. In addition to granting full control to the object owner, the XML specifies full control to an AWS account identified by its canonical user ID.

```
PUT /my-image.jpg?acl HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: authorization string
Content-Length: 124

<AccessControlPolicy>
  <Owner>
    <ID>75aa57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
    <DisplayName>CustomersName@amazon.com</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
```

```
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="CanonicalUser">
        <ID>75aa57f09aa0c8caeab4f8c24e99d10f8e7faeeExampleCanonicalUserID</ID>

        <DisplayName>CustomerName@amazon.com</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

Sample Response

The following shows a sample response when versioning on the bucket is enabled.

```
HTTP/1.1 200 OK
x-amz-id-2: eftixk72aD6Ap51T9AS1ed4OpIszj7UDNEHGran
x-amz-request-id: 318BC8BC148832E5
x-amz-version-id: 3/L4kqtJlcpXrof3vjVBH40Nr8X8gdRQBpUMLUo
Date: Wed, 28 Oct 2009 22:32:00 GMT
Last-Modified: Sun, 1 Jan 2006 12:00:00 GMT
Content-Length: 0
Connection: close
Server: AmazonS3
```

Sample Request: Setting the ACL of a specified object version

The following request sets the ACL on the specified version of the object.

```
PUT /my-image.jpg?acl&versionId=3HL4kqtJlcpXroDTDmJ+rmSpXd3dIb
rHY+MTRCxf3vjVBH40Nrjfkd HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: authorization string
Content-Length: 124

<AccessControlPolicy>
  <Owner>
    <ID>75aa57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
    <DisplayName>mtd@amazon.com</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="CanonicalUser">
       <ID>75aa57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>

        <DisplayName>mtd@amazon.com</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-id-2: eftixk72aD6Ap51u8yU9AS1ed4OpIszj7UDNEHGran
x-amz-request-id: 318BC8BC148832E5
x-amz-version-id: 3/L4kqtJlcpXro3vjVBH40Nr8X8gdRQBpUMLUo
Date: Wed, 28 Oct 2009 22:32:00 GMT
Last-Modified: Sun, 1 Jan 2006 12:00:00 GMT
Content-Length: 0
Connection: close
Server: AmazonS3
```

Sample Request: Access permissions specified using headers

The following request uses ACL-specific request headers, *x-amz-acl*, and specifies a canned ACL (public_read) to grant object read access to everyone.

```
PUT ExampleObject.txt?acl HTTP/1.1
Host: examplebucket.s3.amazonaws.com
x-amz-acl: public-read
Accept: */*
Authorization: authorization string
Host: s3.amazonaws.com
Connection: Keep-Alive
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-id-2: w5YegkbG6ZDsje4WK56RWPxNQHIQ0CjrjyRVFZhEJI9E3kbabXnBO9w5G7Dmxsgk
x-amz-request-id: C13B2827BD8455B1
Date: Sun, 29 Apr 2012 23:24:12 GMT
Content-Length: 0
Server: AmazonS3
```

Related Resources

- PUT Object - Copy (p. 269)
- POST Object (p. 238)
- GET Object (p. 212)

PUT Object - Copy

Description

This implementation of the PUT operation creates a copy of an object that is already stored in Amazon S3. A PUT copy operation is the same as performing a GET and then a PUT. Adding the request header, *x-amz-copy-source*, makes the PUT operation copy the source object into the destination bucket.

> **Note**
> You can store individual objects of up to 5 TB in Amazon S3. You create a copy of your object up to 5 GB in size in a single atomic operation using this API. However, for copying an object greater than 5 GB, you must use the multipart upload API. For conceptual information on multipart upload, go to Uploading Objects Using Multipart Upload in the *Amazon Simple Storage Service Developer Guide*.

When copying an object, you can preserve most of the metadata (default) or specify new metadata. However, the ACL is not preserved and is set to `private` for the user making the request.

All copy requests must be authenticated and cannot contain a message body. Additionally, you must have READ access to the source object and WRITE access to the destination bucket. For more information, see REST Authentication.

To copy an object only under certain conditions, such as whether the `ETag` matches or whether the object was modified before or after a specified date, use the request headers `x-amz-copy-source-if-match`, `x-amz-copy-source-if-none-match`, `x-amz-copy-source-if-unmodified-since`, or `x-amz-copy-source-if-modified-since`.

> **Note**
> All headers prefixed with `x-amz-` must be signed, including `x-amz-copy-source`.

You can use this operation to change storage class of an object that is already stored in Amazon S3 using the `x-amz-storage-class` request header. For more information, go to Changing the Storage Class of an Object in Amazon S3 in the *Amazon Simple Storage Service Developer Guide*.

The source object that you are copying can be encrypted or unencrypted. If the source object is encrypted, it can be encrypted by server-side encryption using AWS-managed encryption keys or by using a customer-provided encryption key. When copying an object you can request that Amazon S3 encrypt the target object by using either the AWS-managed encryption keys or by using your own encryption key, regardless of what form of server-side encryption was used to encrypt the source or if the source object was not encrypted. For more information about server-side encryption, go to Using Server-Side Encryption in the *Amazon Simple Storage Service Developer Guide*.

There are two opportunities for a copy request to return an error. One can occur when Amazon S3 receives the copy request and the other can occur while Amazon S3 is copying the files. If the error occurs before the `copy` operation starts, you receive a standard Amazon S3 error. If the error occurs during the `copy` operation, the error response is embedded in the `200 OK` response. This means that a `200 OK` response can contain either a success or an error. Make sure to design your application to parse the contents of the response and handle it appropriately.

If the copy is successful, you receive a response that contains the information about the copied object.

> **Note**
> If the request is an HTTP 1.1 request, the response is chunk encoded. Otherwise, it will not contain the content-length and you will need to read the entire body.

Versioning

By default, *x-amz-copy-source* identifies the current version of an object to copy. (If the current version is a delete marker, Amazon S3 behaves as if the object was deleted.) To copy a different version, use the *versionId* subresource.

If you enable versioning on the target bucket, Amazon S3 generates a unique version ID for the object being copied. This version ID is different from the version ID of the source object. Amazon S3 returns the version ID of the copied object in the *x-amz-version-id* response header in the response.

If you do not enable versioning or suspend it on the target bucket, the version ID Amazon S3 generates is always `null`.

If the source object's storage class is `GLACIER`, then you must first restore a copy of this object before you can use it as a source object for the copy operation. For more information, see POST Object restore (p. 247).

To see sample requests that use versioning, see Sample Request: Copying a specified version of an object (p. 280).

Access Permissions

When copying an object, you can optionally specify the accounts or groups that should be granted specific permissions on the new object. There are two ways to grant the permissions using the request headers:

- Specify a canned ACL using the `x-amz-acl` request header. For more information, see Canned ACL in the *Amazon Simple Storage Service Developer Guide*.
- Specify access permissions explicitly using the `x-amz-grant-read`, `x-amz-grant-read-acp`, `x-amz-grant-write-acp`, and `x-amz-grant-full-control` headers. These headers map to the set of permissions Amazon S3 supports in an ACL. For more information, go to Access Control List (ACL) Overview in the *Amazon Simple Storage Service Developer Guide*.

> **Note**
> You can use either a canned ACL or specify access permissions explicitly. You cannot do both.

Requests

Syntax

```
PUT /destinationObject HTTP/1.1
Host: destinationBucket.s3.amazonaws.com
x-amz-copy-source: /source_bucket/sourceObject
x-amz-metadata-directive: metadata_directive
x-amz-copy-source-if-match: etag
x-amz-copy-source-if-none-match: etag
x-amz-copy-source-if-unmodified-since: time_stamp
x-amz-copy-source-if-modified-since: time_stamp
<request metadata>
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))
Date: date
```

Note
The syntax shows only some of the request headers. For a complete list, see the Request Headers section.

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

This implementation of the operation can use the following request headers in addition to the request headers common to all operations. For more information, see Common Request Headers (p. 12).

Name	Description	Required
x-amz-copy-source	The name of the source bucket and key name of the source object, separated by a slash (/). Type: String Default: None Constraints: This string must be URL-encoded. Additionally, the source bucket must be valid and you must have READ access to the valid source object. If the source object is archived in Amazon Glacier (storage class of the object is `GLACIER`), you must first restore a temporary copy using the POST Object restore (p. 247). Otherwise, Amazon S3 returns the `403 ObjectNotInActiveTierError error` response.	Yes

Name	Description	Required
x-amz-metadata-directive	Specifies whether the metadata is copied from the source object or replaced with metadata provided in the request. • If copied, the metadata, except for the version ID, remains unchanged. In addition, the `server-side-encryption`, `storage-class`, and `website-redirect-location` metadata from the source is not copied. If you specify this metadata explicitly in the copy request, Amazon S3 adds this metadata to the resulting object. If you specify headers in the request specifying any user-defined metadata, Amazon S3 ignores these headers. • If replaced, all original metadata is replaced by the metadata you specify. Type: String Default: `COPY` Valid values: `COPY` | `REPLACE` Constraints: Values other than `COPY` or `REPLACE` result in an immediate 400-based error response. You cannot copy an object to itself unless the `MetadataDirective` header is specified and its value set to `REPLACE`. For information on supported metadata, see Common Request Headers (p. 12)	No
x-amz-copy-source-if-match	Copies the object if its entity tag (`ETag`) matches the specified tag; otherwise, the request returns a 412 HTTP status code error (failed precondition). Type: String Default: None Constraints: This header can be used with `x-amz-copy-source-if-unmodified-since`, but cannot be used with other conditional copy headers.	No
x-amz-copy-source-if-none-match	Copies the object if its entity tag (`ETag`) is different than the specified `ETag`; otherwise, the request returns a 412 HTTP status code error (failed precondition). Type: String Default: None Constraints: This header can be used with `x-amz-copy-source-if-modified-since`, but cannot be used with other conditional copy headers.	No

Name	Description	Required
x-amz-copy-source-if-unmodified-since	Copies the object if it hasn't been modified since the specified time; otherwise, the request returns a 412 HTTP status code error (failed precondition). Type: String Default: None Constraints: This must be a valid HTTP date. For more information, go to http://www.ietf.org/rfc/rfc2616.txt. This header can be used with `x-amz-copy-source-if-match`, but cannot be used with other conditional copy headers.	No
x-amz-copy-source-if-modified-since	Copies the object if it has been modified since the specified time; otherwise, the request returns a 412 HTTP status code error (failed condition). Type: String Default: None Constraints: This must be a valid HTTP date. This header can be used with `x-amz-copy-source-if-none-match`, but cannot be used with other conditional copy headers.	No
x-amz-storage-class	RRS enables customers to reduce their costs by storing noncritical, reproducible data at lower levels of redundancy than Amazon S3's standard storage. Type: Enum Default: `STANDARD` Valid Values: `STANDARD` | `REDUCED_REDUNDANCY` Constraints: You cannot specify `GLACIER` as the storage class. To transition objects to the GLACIER storage class you can use lifecycle configuration.	No

Name	Description	Required

x-amz-website-redirect-location	If the bucket is configured as a website, redirects requests for this object to another object in the same bucket or to an external URL. Amazon S3 stores the value of this header in the object metadata. For information about object metadata, go to Object Key and Metadata.
	
	In the following example, the request header sets the redirect to an object (anotherPage.html) in the same bucket:
	`x-amz-website-redirect-location: /anotherPage.html`
	
	In the following example, the request header sets the object redirect to another website:
	
	`x-amz-website-redirect-location: http://www.example.com/`
	
	For more information about website hosting in Amazon S3, go to sections Hosting Websites on Amazon S3 and How to Configure Website Page Redirects in the *Amazon Simple Storage Service Developer Guide*.
	Type: String
	Default: None
	Constraints: The value must be prefixed by, "/", "http://" or "https://". The length of the value is limited to 2 K.
		No

Server-Side Encryption Specific Request Headers

If you want your target object encrypted, you will need to provide appropriate encryption related request headers depending on whether you want to use AWS-managed encryption keys or provide your own encryption key:

- If you want the target object encrypted using server-side encryption with an AWS-managed encryption key, you provide the following request header.

Name	Description	Required
x-amz-server-side-encryption	Specifies the server-side encryption algorithm to use when Amazon S3 creates the target object. Type: String Valid Value: AES256	Yes

- If you want the target object encrypted using server-side encryption with an encryption key you provide, you must provide encryption information using the following headers.

Name	Description	Required
x-amz-server-side-encryption-customer-algorithm	Specifies the algorithm to use to when encrypting the object. Type: String Default: None Valid Value: AES256 Constraints: Must be accompanied by valid *x-amz-server-side-encryption-customer-key* and *x-amz-server-side-encryption-customer-key-MD5* headers.	Yes
x-amz-server-side-encryption-customer-key	Specifies the customer provided base-64 encoded encryption key for Amazon S3 to use in encrypting data. This value is used to store the object and then is discarded; Amazon does not store the encryption key. The key must be appropriate for use with the algorithm specified in the *x-amz-server-side-encryption-customer-algorithm* header. Type: String Default: None Constraints: Must be accompanied by valid *x-amz-server-side-encryption-customer-algorithm* and *x-amz-server-side-encryption-customer-key-MD5* headers.	Yes
x-amz-server-side-encryption-customer-key-MD5	Specifies the base64-encoded 128-bit MD5 digest of the encryption key according to RFC 1321. Amazon S3 uses this header as a message integrity check to ensure the encryption key was transmitted without error. Type: String Default: None Constraints: Must be accompanied by valid *x-amz-server-side-encryption-customer-algorithm* and *x-amz-server-side-encryption-customer-key* headers.	Yes

- If the source object is encrypted using server-side encryption with customer-provided encryption keys, you must use the following headers providing encryption information so that Amazon S3 can decrypt the object for copying.

Name	Description	Required
x-amz-copy-source-server-side-encryption-customer-algorithm	Specifies algorithm to use when decrypting the source object. Type: String Default: None Valid Value: AES256 Constraints: Must be accompanied by valid *x-amz-copy-source-server-side-encryption-customer-key* and *x-amz-copy-source-server-side-encryption-customer-key-MD5* headers.	Yes
x-amz-copy-source-server-side-encryption-customer-key	Specifies the customer provided base-64 encoded encryption key for Amazon S3 to use to decrypt the source object. After the copy operation, Amazon S3 will discard this key. The encryption key provided in this header must be one that was used when the source object was created. Type: String Default: None Constraints: Must be accompanied by valid *x-amz-copy-source-server-side-encryption-customer-algorithm* and *x-amz-copy-source-server-side-encryption-customer-key-MD5* headers.	Yes
x-amz-copy-source-server-side-encryption-customer-key-MD5	Specifies the base64-encoded 128-bit MD5 digest of the encryption key according to RFC 1321. Amazon S3 uses this header for a message integrity check to ensure the encryption key was transmitted without error. Type: String Default: None Constraints: Must be accompanied by valid *x-amz-copy-source-server-side-encryption-customer-algorithm* and *x-amz-copy-source-server-side-encryption-customer-key* headers.	Yes

Access Control List (ACL) Specific Request Headers

Additionally, you can use the following access control–related headers with this operation. By default, all objects are private; only the owner has full access control. When adding a new object, you can grant permissions to individual AWS accounts or predefined groups defined by Amazon S3. These permissions are then added to the Access Control List (ACL) on the object. For more information, go to Using ACLs. This operation enables you grant access permissions using one of the following two methods:

- **Specify a canned ACL —** Amazon S3 supports a set of predefined ACLs, known as canned ACLs. Each canned ACL has a predefined set of grantees and permissions. For more information, go to Canned ACL.

Name	Description	Required
x-amz-acl	The canned ACL to apply to the object. Type: String Default: private Valid Values: `private | public-read | public-read-write | authenticated-read | bucket-owner-read | bucket-owner-full-control` Constraints: None	No

- **Specify access permissions explicitly —** If you want to explicitly grant access permissions to specific AWS accounts or groups, you can use the following headers. Each of these headers maps to specific permissions Amazon S3 supports in an ACL. For more information, go to Access Control List (ACL) Overview. In the header, you specify a list of grantees who get the specific permission.

Name	Description	Required
x-amz-grant-read	Allows grantee to read the object data and its metadata. Type: String Default: None Constraints: None	No
x-amz-grant-write	Not applicable. This applies only when granting access permissions on a bucket. Type: String Default: None Constraints: None	No
x-amz-grant-read-acp	Allows grantee to read the object ACL. Type: String Default: None Constraints: None	No
x-amz-grant-write-acp	Allows grantee to write the ACL for the applicable object. Type: String Default: None Constraints: None	No
x-amz-grant-full-control	Allows grantee the READ, READ_ACP, and WRITE_ACP permissions on the object. Type: String Default: None Constraints: None	No

You specify each grantee as a `type=value` pair, where the type can be one of the following:

- **emailAddress** – if value specified is the email address of an AWS account
- **id** – if value specified is the canonical user ID of an AWS account

- **uri** – if granting permission to a predefined group.

For example, the following `x-amz-grant-read` header grants read object data and its metadata permission to the AWS accounts identified by their email addresses.

```
x-amz-grant-read: emailAddress="xyz@amazon.com", emailAddress="abc@amazon.com"
```

Request Elements

This implementation of the operation does not use request elements.

Responses

Response Headers

This implementation of the operation can include the following response headers in addition to the response headers common to all responses. For more information, see Common Response Headers (p. 14).

Name	Description
x-amz-expiration	Amazon S3 will return this header if an `Expiration` action is configured for the object as part of the bucket's lifecycle configuration. The header value includes an "expiry-date" component and a URL-encoded "rule-id" component. Note that for version-enabled buckets, this header applies only to current versions; Amazon S3 does not provide a header to infer when a noncurrent version will be eligible for permanent deletion. For more information, see PUT Bucket lifecycle (p. 162). Type: String
x-amz-copy-source-version-id	Version of the source object that was copied. Type: String
x-amz-server-side-encryption	If you request server-side encryption, the response includes this header confirming the encryption algorithm used for the target object. Type: String
x-amz-server-side -encryption -customer-algorithm	If server-side encryption with customer-provided encryption keys (SSE-C) encryption was requested the response will include this header confirming the encryption algorithm used for the destination object. Type: String Valid Values: `AES256`
x-amz-server-side -encryption -customer-key-MD5	If SSE-C encryption was requested, the response includes this header to provide roundtrip message integrity verification of the customer-provided encryption key used to encrypt the destination object. Type: String
x-amz-version-id	Version of the copied object in the destination bucket. Type: String

Response Elements

Name	Description
CopyObjectResult	Container for all response elements. Type: Container Ancestor: None
ETag	Returns the ETag of the new object. The ETag reflects changes only to the contents of an object, not its metadata. Type: String Ancestor: CopyObjectResult
LastModified	Returns the date the object was last modified. Type: String Ancestor: CopyObjectResult

Special Errors

This implementation of the operation does not return special errors. For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

Sample Request

This example copies my-image.jpg into the bucket, bucket, with the key name my-second-image.jpg.

```
PUT /my-second-image.jpg HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
x-amz-copy-source: /bucket/my-image.jpg
Authorization: authorization string
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-id-2: eftixk72aD6Ap51TnqcoF8eFidJG9Z/2mkiDFu8yU9AS1ed4OpIszj7UDNEHGran
x-amz-request-id: 318BC8BC148832E5
x-amz-copy-source-version-id: 3/L4kqtJlcpXroDTDmJ+rmSpXd3dIb
rHY+MTRCxf3vjVBH40Nr8X8gdRQBpUMLUo
x-amz-version-id: QUpfdndhfd8438MNFDN93jdnJFkdmqnh893
Date: Wed, 28 Oct 2009 22:32:00 GMT
Connection: close
Server: AmazonS3

<CopyObjectResult>
   <LastModified>2009-10-28T22:32:00</LastModified>
   <ETag>"9b2cf535f27731c974343645a3985328"</ETag>
 </CopyObjectResult>
```

x-amz-version-id returns the version ID of the object in the destination bucket, and *x-amz-copy-source-version-id* returns the version ID of the source object.

Sample Request: Copying a specified version of an object

The following request copies the key *my-image.jpg* with the specified version ID and copies it into the bucket *bucket* and gives it the key *my-second-image.jpg*.

```
PUT /my-second-image.jpg HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
x-amz-copy-source: /bucket/my-image.jpg?versionId=3/L4kqtJlcpXroDTDmJ+rmSpXd3dIb
rHY+MTRCxf3vjVBH40Nr8X8gdRQBpUMLUo
Authorization: authorization string
```

Success Response: Copying a versioned object into a version enabled bucket

The following response shows that an object was copied into a target bucket where Versioning is enabled.

```
HTTP/1.1 200 OK
x-amz-id-2: eftixk72aD6Ap51TnqcoF8eFidJG9Z/2mkiDFu8yU9AS1ed4OpIszj7UDNEHGran
x-amz-request-id: 318BC8BC148832E5
x-amz-version-id: QUpfdndhfd8438MNFDN93jdnJFkdmqnh893
 x-amz-copy-source-version-id: 09df8234529fjs0dfi0w52935029wefdj
Date: Wed, 28 Oct 2009 22:32:00 GMT
Connection: close
Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<CopyObjectResult>
   <LastModified>2009-10-28T22:32:00</LastModified>
   <ETag>"9b2cf535f27731c974343645a3985328"</ETag>
</CopyObjectResult>
```

Success Response: Copying a versioned object into a version suspended bucket

The following response shows that an object was copied into a target bucket where versioning is suspended. Note that the parameter <*VersionId*> does not appear.

```
HTTP/1.1 200 OK
x-amz-id-2: eftixk72aD6Ap51TnqcoF8eFidJG9Z/2mkiDFu8yU9AS1ed4OpIszj7UDNEHGran
x-amz-request-id: 318BC8BC148832E5
x-amz-copy-source-version-id: 3/L4kqtJlcpXroDTDmJ+rmSpXd3dIb
rHY+MTRCxf3vjVBH40Nr8X8gdRQBpUMLUo
Date: Wed, 28 Oct 2009 22:32:00 GMT
Connection: close
Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<CopyObjectResult>
  <LastModified>2009-10-28T22:32:00</LastModified>
  <ETag>"9b2cf535f27731c974343645a3985328"</ETag>
</CopyObjectResult>
```

Sample: Copy from non-encrypted object to an object encrypted with server-side encryption with customer-provided encryption keys

The following example specifies the HTTP PUT header to copy a non-encrypted object to an object encrypted with server-side encryption with customer-provided encryption keys (SSE-C).

```
PUT /exampleDestinationObject HTTP/1.1
Host: example-destination-bucket.s3.amazonaws.com
x-amz-server-side-encryption-customer-algorithm: AES256
x-amz-server-side-encryption-customer-key: Base64(YourKey)
x-amz-server-side-encryption-customer-key-MD5 : Base64(MD5(YourKey))
x-amz-metadata-directive: metadata_directive
x-amz-copy-source: /example_source_bucket/exampleSourceObject
x-amz-copy-source-if-match: etag
x-amz-copy-source-if-none-match: etag
x-amz-copy-source-if-unmodified-since: time_stamp
x-amz-copy-source-if-modified-since: time_stamp
<request metadata>
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))
Date: date
```

Sample: Copy from an object encrypted with SSE-C to an object encrypted with SSE-C

The following example specifies the HTTP PUT header to copy an object encrypted with server-side encryption with customer-provided encryption keys to an object encrypted with server-side encryption with customer-provided encryption keys for key rotation.

```
PUT /exampleDestinationObject HTTP/1.1
Host: example-destination-bucket.s3.amazonaws.com
x-amz-server-side-encryption-customer-algorithm: AES256
x-amz-server-side-encryption-customer-key: Base64(NewKey)
x-amz-server-side-encryption-customer-key-MD5: Base64(MD5(NewKey))
x-amz-metadata-directive: metadata_directive
x-amz-copy-source: /source_bucket/sourceObject
x-amz-copy-source-if-match: etag
x-amz-copy-source-if-none-match: etag
x-amz-copy-source-if-unmodified-since: time_stamp
x-amz-copy-source-if-modified-since: time_stamp
x-amz-copy-source-server-side-encryption-customer-algorithm: AES256
x-amz-copy-source-server-side-encryption-customer-key: Base64(OldKey)
x-amz-copy-source-server-side-encryption-customer-key-MD5: Base64(MD5(OldKey))
<request metadata>
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))
Date: date
```

Related Resources

- Copying Objects
- PUT Object (p. 250)
- GET Object (p. 212)

Initiate Multipart Upload

Description

This operation initiates a multipart upload and returns an upload ID. This upload ID is used to associate all the parts in the specific multipart upload. You specify this upload ID in each of your subsequent upload part requests (see Upload Part (p. 290)). You also include this upload ID in the final request to either complete or abort the multipart upload request.

For more information on multipart uploads, go to Multipart Upload Overview in the *Amazon Simple Storage Service Developer Guide*.

For information on permissions required to use the multipart upload API, go to Multipart Upload API and Permissions in the *Amazon Simple Storage Service Developer Guide*.

For request signing, multipart upload is just a series of regular requests, you initiate multipart upload, send one or more requests to upload parts, and finally complete multipart upload. You sign each request individually, there is nothing special about signing multipart upload requests. For more information about signing, see Authenticating Requests (AWS Signature Version 4) (p. 15).

> **Note**
> After you initiate multipart upload and upload one or more parts, you must either complete or abort multipart upload in order to stop getting charged for storage of the uploaded parts. Only after you either complete or abort multipart upload, Amazon S3 frees up the parts storage and stops charging you for the parts storage.

You can optionally request server-side encryption where Amazon S3 encrypts your data as it writes it to disks in its data centers and decrypts it for you when you access it. You have the option to provide your own encryption key or use AWS-managed encryption keys. If you choose to provide your own encryption key, the relevant request headers you provide in this initiate request must also appear in the subsequent requests you send to upload parts. For more information, go to Using Server-Side Encryption in the *Amazon Simple Storage Service Developer Guide*.

Requests

Syntax

```
POST /ObjectName?uploads HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: authorization string (see Authenticating Requests (AWS Signature
 Version 4) (p. 15))
```

Request Parameters

This operation does not use request parameters.

Request Headers

Name	Description	Required
Cache-Control	Can be used to specify caching behavior along the request/reply chain. For more information, go to http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.9. Type: String Default: None	No
Content-Disposition	Specifies presentational information for the object. For more information, go to http://www.w3.org/Protocols/rfc2616/rfc2616-sec19.html#sec19.5.1. Type: String Default: None	No
Content-Encoding	Specifies what content encodings have been applied to the object and thus what decoding mechanisms must be applied to obtain the media-type referenced by the *Content-Type* header field. For more information, go to http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.11. Type: String Default: None	No
Content-Type	A standard MIME type describing the format of the object data. For more information, go to http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.17. Type: String Default: `binary/octel-stream` Constraints: MIME types only	No
Expires	The date and time at which the object is no longer cacheable. For more information, go to http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.21. Type: String Default: None	No
x-amz-meta-	Any header starting with this prefix is considered user metadata. It will be stored with the object and returned when you retrieve the object. Type: String Default: None	No
x-amz-storage-class	The type of storage to use for the object that is created after successful multipart upload. Type: String Valid Values: `STANDARD` | `REDUCED_REDUNDANCY` Default: `STANDARD` Constraints: You cannot specify `GLACIER` as the storage class. To transition objects to the GLACIER storage class you can use lifecycle configuration.	No

Name	Description	Required

x-amz-website-redirect-location	If the bucket is configured as a website, redirects requests for this object to another object in the same bucket or to an external URL. Amazon S3 stores the value of this header in the object metadata. For information about object metadata, go to Object Key and Metadata.
	
	In the following example, the request header sets the redirect to an object (anotherPage.html) in the same bucket:
	`x-amz-website-redirect-location: /anotherPage.html`
	
	In the following example, the request header sets the object redirect to another website:
	
	`x-amz-website-redirect-location: http://www.example.com/`
	
	For more information about website hosting in Amazon S3, go to sections Hosting Websites on Amazon S3 and How to Configure Website Page Redirects in the *Amazon Simple Storage Service Developer Guide*.
	Type: String
	Default: None
	Constraints: The value must be prefixed by, "/", "http://" or "https://". The length of the value is limited to 2 K.
		No

Access Control List (ACL) Specific Request Headers

Additionally, you can use the following access control-related headers with this operation. By default, all objects are private, only the owner has full access control. When adding a new object, you can grant permissions to individual AWS accounts or predefined groups defined by Amazon S3. These permissions are then added to the Access Control List (ACL) on the object. For more information, go to Access Control List (ACL) Overview in the *Amazon Simple Storage Service Developer Guide*. This operation enables you to grant access permissions using one of the following two ways:

- **Specify canned ACL** – Amazon S3 supports a set of predefined ACLs, known as canned ACLs. Each canned ACL has a predefined set of grantees and permissions. For more information, go to Canned ACL.

Name	Description	Required
x-amz-acl	The canned ACL to apply to the object. Type: String Default: private Valid Values: `private | public-read | public-read-write | authenticated-read | bucket-owner-read | bucket-owner-full-control` Constraints: None	No

- **Specify access permissions explicitly** – If you want to explicitly grant access permissions to specific AWS accounts or groups, you can use the following headers. Each of these headers maps to specific

permissions Amazon S3 supports in an ACL. For more information, go to Access Control List (ACL) Overview. In the header, you specify a list of grantees who get the specific permission.

Name	Description	Required
x-amz-grant-read	Allows grantee to read the object data and its metadata. Type: String Default: None Constraints: None	No
x-amz-grant-write	Not applicable. Type: String Default: None Constraints: None	No
x-amz-grant-read-acp	Allows grantee to read the object ACL. Type: String Default: None Constraints: None	No
x-amz-grant-write-acp	Allows grantee to write the ACL for the applicable object. Type: String Default: None Constraints: None	No
x-amz-grant-full-control	Allows grantee the READ, READ_ACP, and WRITE_ACP permissions on the object. Type: String Default: None Constraints: None	No

You specify each grantee as a `type=value` pair, where the type can be one of the following::

- **emailAddress** – if value specified is the email address of an AWS account
- **id** – if value specified is the canonical user ID of an AWS account
- **uri** – if granting permission to a predefined group.

For example, the following `x-amz-grant-read` header grants read object data and its metadata permission to the AWS accounts identified by their email addresses.

```
x-amz-grant-read: emailAddress="xyz@amazon.com", emailAddress="abc@amazon.com"
```

Server-Side Encryption Specific Request Headers

You can optionally request Amazon S3 to encrypt data at rest using server-side encryption. Server-side encryption is about data encryption at rest, that is, Amazon S3 encrypts your data as it writes it to disks in its data centers and decrypts it for you when you access it. Depending on whether you want to use AWS-managed encryption keys or provide your own encryption keys, you use the following headers:

- Use AWS-managed encryption keys — If you want Amazon S3 to manage keys used to encrypt data, you specify the following header in the request.

Name	Description	Required
x-amz-server-side-encryption	Specifies a server-side encryption algorithm to use when Amazon S3 creates an object. Type: String Valid Value: AES256	Yes

- Use customer-provided encryption keys — If you want to manage your own encryption keys, you must provide all the following headers in the request.

Name	Description	Required
x-amz-server-side-encryption-customer-algorithm	Specifies the algorithm to use to when encrypting the object. Type: String Default: None Valid Value: AES256 Constraints: Must be accompanied by valid *x-amz-server-side-encryption-customer-key* and *x-amz-server-side-encryption-customer-key-MD5* headers.	Yes
x-amz-server-side-encryption-customer-key	Specifies the customer provided base-64 encoded encryption key for Amazon S3 to use in encrypting data. This value is used to store the object and then is discarded; Amazon does not store the encryption key. The key must be appropriate for use with the algorithm specified in the *x-amz-server-side-encryption-customer-algorithm* header. Type: String Default: None Constraints: Must be accompanied by valid *x-amz-server-side-encryption-customer-algorithm* and *x-amz-server-side-encryption-customer-key-MD5* headers.	Yes
x-amz-server-side-encryption-customer-key-MD5	Specifies the base64-encoded 128-bit MD5 digest of the encryption key according to RFC 1321. Amazon S3 uses this header for message integrity check to ensure the encryption key was transmitted without error. Type: String Default: None Constraints: Must be accompanied by valid *x-amz-server-side-encryption-customer-algorithm* and *x-amz-server-side-encryption-customer-key* headers.	Yes

Request Elements

This operation does not use request elements.

Responses

Response Headers

This implementation of the operation can include the following response headers in addition to the response headers common to all responses. For more information, see Common Response Headers (p. 14).

Name	Description
x-amz-server-side-encryption	If you specify server-side encryption in your request, the response includes this header. It confirms the encryption algorithm that will be used for the object that is created after successful multipart upload. Type: String
x-amz-server-side-encryption-customer-algorithm	If server-side encryption with customer-provided encryption keys encryption was requested, the response will include this header confirming the encryption algorithm used. Type: String Valid Values: `AES256`
x-amz-server-side-encryption-customer-key-MD5	If server-side encryption using customer-provided encryption key was requested, the response returns this header to provide roundtrip message integrity verification of the customer-provided encryption key. Type: String

Response Elements

Name	Description
`InitiateMultipartUploadResult`	Container for response. Type: Container Children: `Bucket`, `Key`, `UploadId` Ancestors: None
`Bucket`	Name of the bucket to which the multipart upload was initiated. Type: string Ancestors: `InitiateMultipartUploadResult`
`Key`	Object key for which the multipart upload was initiated. Type: String Ancestors: `InitiateMultipartUploadResult`
`UploadId`	ID for the initiated multipart upload. Type: String Ancestors: `InitiateMultipartUploadResult`

Special Errors

This implementation of the operation does not return special errors. For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

Sample Request

This operation initiates a multipart upload for the `example-object` object.

```
POST /example-object?uploads HTTP/1.1
Host: example-bucket.s3.amazonaws.com
Date: Mon, 1 Nov 2010 20:34:56 GMT
Authorization: authorization string
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-id-2: Uuag1LuByRx9e6j5Onimru9pO4ZVKnJ2Qz7/C1NPcfTWAtRPfTaOFg==
x-amz-request-id: 656c76696e6727732072657175657374
Date:  Mon, 1 Nov 2010 20:34:56 GMT
Content-Length: 197
Connection: keep-alive
Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<InitiateMultipartUploadResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">

  <Bucket>example-bucket</Bucket>
  <Key>example-object</Key>
  <UploadId>VXBsb2FkIElEIGZvciA2aWWpbmcncyBteS1tb3ZpZS5tMnRzIHVwbG9hZA</UploadId>
</InitiateMultipartUploadResult>
```

Sample: Initiate multipart upload, using server-side encryption with customer-provided encryption keys

This example initiate multipart upload request specifies server-side encryption with customer-provided encryption keys by adding relevant headers.

```
POST /example-object?uploads HTTP/1.1
Host: example-bucket.s3.amazonaws.com
Authorization:authorization string
Date: Wed, 28 May 2014 19:34:57 +0000
x-amz-server-side-encryption-customer-key:
g0lCfA3Dv40jZz5SQJ1ZukLRFqtI5WorC/8SEEXAMPLE
x-amz-server-side-encryption-customer-key-MD5: ZjQrne1X/iTcskbY2example
x-amz-server-side-encryption-customer-algorithm: AES256
```

In the response, Amazon S3 returns an UploadId. In addition, Amazon S3 returns the encryption algorithm and the MD5 digest of the encryption key you provided in the request.

```
HTTP/1.1 200 OK
x-amz-id-2: 36HRCaIGp57F1FvWvVRrvd3hNn9WoBGfEaCVHTCt8QWf00qxdHazQUgfoXAbhFWD

x-amz-request-id: 50FA1D691B62CA43
Date: Wed, 28 May 2014 19:34:58 GMT
x-amz-server-side-encryption-customer-algorithm: AES256
x-amz-server-side-encryption-customer-key-MD5: ZjQrne1X/iTcskbY2m3tFg==
Transfer-Encoding: chunked

<?xml version="1.0" encoding="UTF-8"?>
<InitiateMultipartUploadResult
xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
   <Bucket>example-bucket</Bucket>
   <Key>example-object</Key>
   <UploadId>EXAMPLEJZ6e0YupT2h66iePQCc9IEbYbDUy4RTpMeoSMLPRp8Z5o1u8feSRonpvn
WsKKG35tI2LB9VDPiCgTy.Gq2VxQLYjrue4Nq.NBdqI-</UploadId>
</InitiateMultipartUploadResult>
```

Related Actions

- Upload Part (p. 290)
- Complete Multipart Upload (p. 302)
- Abort Multipart Upload (p. 308)
- List Parts (p. 310)
- List Multipart Uploads (p. 135)

Upload Part

Description

This operation uploads a part in a multipart upload.

> **Note**
> In this operation, you provide part data in your request. However, you have an option to specify your existing Amazon S3 object as a data source for the part you are uploading. To upload a part from an existing object, you use the Upload Part (Copy) operation. For more information, see Upload Part - Copy (p. 295).

You must initiate a multipart upload (see Initiate Multipart Upload (p. 282)) before you can upload any part. In response to your initiate request, Amazon S3 returns an upload ID, a unique identifier, that you must include in your upload part request.

Part numbers can be any number from 1 to 10,000, inclusive. A part number uniquely identifies a part and also defines its position within the object being created. If you upload a new part using the same part number that was used with a previous part, the previously uploaded part is overwritten. Each part must be at least 5 MB in size, except the last part. There is no size limit on the last part of your multipart upload.

To ensure that data is not corrupted when traversing the network, specify the `Content-MD5` header in the upload part request. Amazon S3 checks the part data against the provided MD5 value. If they do not match, Amazon S3 returns an error.

> **Note**
> After you initiate multipart upload and upload one or more parts, you must either complete or abort multipart upload in order to stop getting charged for storage of the uploaded parts. Only after you either complete or abort the multipart upload, Amazon S3 frees up the parts storage and stops charging you for it.

For more information on multipart uploads, go to Multipart Upload Overview in the *Amazon Simple Storage Service Developer Guide* .

For information on the permissions required to use the multipart upload API, go to Multipart Upload API and Permissions in the *Amazon Simple Storage Service Developer Guide*.

You can optionally request server-side encryption where Amazon S3 encrypts your data as it writes it to disks in its data centers and decrypts it for you when you access it. You have the option of providing your own encryption key, or you can use the AWS-managed encryption keys. If you choose to provide your own encryption key, the request headers you provide in the request must match the headers you used in the request to initiate the upload by using Initiate Multipart Upload (p. 282). For more information, go to Using Server-Side Encryption in the *Amazon Simple Storage Service Developer Guide*.

Requests

Syntax

```
PUT /ObjectName?partNumber=PartNumber&uploadId=UploadId HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Content-Length: Size
Authorization: authorization string
```

Request Parameters

This operation does not use request parameters.

Request Headers

This implementation of the operation can use the following request headers in addition to the request headers common to all operations. For more information, see Common Request Headers (p. 12).

Name	Description	Required
Content-Length	The size of the part, in bytes. For more information, go to http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.13. Type: Integer Default: None	Yes
Content-MD5	The base64-encoded 128-bit MD5 digest of the part data. This header can be used as a message integrity check to verify that the part data is the same data that was originally sent. Although it is optional, we recommend using the Content-MD5 mechanism as an end-to-end integrity check. For more information, see RFC 1864. Type: String Default: None	No
Expect	When your application uses 100-continue, it does not send the request body until it receives an acknowledgment. If the message is rejected based on the headers, the body of the message is not sent. For more information, go to RFC 2616. Type: String Default: None Valid Values: 100-continue	No

Server-Side Encryption Specific Request Headers

You can optionally request Amazon S3 to encrypt data at rest using server-side encryption. Server-side encryption is about data encryption at rest; that is, Amazon S3 encrypts your data as it writes it to disks in its data centers and decrypts it for you when you access it. Depending on whether you want to use AWS-managed encryption keys or provide your own encryption keys, you use the following headers:

- Use AWS-managed encryption keys. If you want Amazon S3 to manage keys used to encrypt data, you specify the following header in the request.

Name	Description	Required
x-amz-server-side-encryption	Specifies a server-side encryption algorithm to use when Amazon S3 creates an object. Type: String Valid Value: AES256	Yes

- Use customer-provided encryption keys. If you want to manage your own encryption keys, you must provide all of the following headers in the request.

Name	Description	Required
x-amz-server-side-encryption-customer-algorithm	Specifies the algorithm to use to when encrypting the object. Type: String Default: None Valid Value: AES256 Constraints: Must be accompanied by valid *x-amz-server-side-encryption-customer-key* and *x-amz-server-side-encryption-customer-key-MD5* headers.	Yes
x-amz-server-side-encryption-customer-key	Specifies the customer-provided base64-encoded encryption key for Amazon S3 to use in encrypting data. This value is used to store the object and then is discarded; Amazon does not store the encryption key. The key must be appropriate for use with the algorithm specified in the *x-amz-server-side-encryption-customer-algorithm* header. Type: String Default: None Constraints: Must be accompanied by valid *x-amz-server-side-encryption-customer-algorithm* and *x-amz-server-side-encryption-customer-key-MD5* headers.	Yes
x-amz-server-side-encryption-customer-key-MD5	Specifies the base64-encoded 128-bit MD5 digest of the encryption key according to RFC 1321. Amazon S3 uses this header for a message integrity check to ensure the encryption key was transmitted without error. Type: String Default: None Constraints: Must be accompanied by valid *x-amz-server-side-encryption-customer-algorithm* and *x-amz-server-side-encryption-customer-key* headers.	Yes

Request Elements

This operation does not use request elements.

Responses

Response Headers

This implementation of the operation can include the following response headers in addition to the response headers common to all responses. For more information, see Common Response Headers (p. 14).

Name	Description
x-amz-server-side-encryption	If you specified server-side encryption in your initiate multipart upload request, the response includes this header. It confirms the encryption algorithm that Amazon S3 used to encrypt the part you uploaded. Type: String
x-amz-server-side-encryption-customer-algorithm	If server-side encryption with customer-provided encryption keys(SSE-C) encryption was requested, the response will include this header confirming the encryption algorithm used. Type: String Valid Values: AES256
x-amz-server-side-encryption-customer-key-MD5	If SSE-C encryption was requested, the response includes this header to provide roundtrip message integrity verification of the customer-provided encryption key. Type: String

Response Elements

This operation does not use response elements.

Special Errors

Error Code	Description	HTTP Status Code	SOAP Fault Code Prefix
NoSuchUpload	The specified multipart upload does not exist. The upload ID might be invalid, or the multipart upload might have been aborted or completed.	404 Not Found	Client

For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

Sample Request

The following PUT request uploads a part (part number 1) in a multipart upload. The request includes the upload ID that you get in response to your Initiate Multipart Upload request.

```
PUT /my-movie.m2ts?partNumber=1&uploadId=VCVsb2FkIElEIGZvciBlbZZpbm
cncyBteS1tb3ZpZS5tMnRzIHVwbG9hZR HTTP/1.1
Host: example-bucket.s3.amazonaws.com
Date:  Mon, 1 Nov 2010 20:34:56 GMT
Content-Length: 10485760
Content-MD5: pUNXr/BjKK5G2UKvaRRrOA==
Authorization: authorization string

***part data omitted***
```

Sample Response

The response includes the ETag header. You need to retain this value for use when you send the Complete Multipart Upload request.

```
HTTP/1.1 200 OK
x-amz-id-2: Vvag1LuByRx9e6j5Onimru9pO4ZVKnJ2Qz7/C1NPcfTWAtRPfTaOFg==
x-amz-request-id: 656c76696e6727732072657175657374
Date:  Mon, 1 Nov 2010 20:34:56 GMT
ETag: "b54357faf0632cce46e942fa68356b38"
Content-Length: 0
Connection: keep-alive
Server: AmazonS3
```

Sample: Upload a part with an encryption key in the request for server-side encryption

If you initiated a multipart upload, see Sample: Initiate multipart upload, using server-side encryption with customer-provided encryption keys (p. 288), with a request to save an object using server-side encryption with a customer-provided encryption key, each part upload must also include the same set of encryption-specific headers as shown in the following example request.

```
PUT /example-object?partNumber=1&uploadId=EXAMPLEJZ6e0YupT2h66iePQCc9IEbYbDUy4RT
pMeoSMLPRp8Z5o1u8feSRonpvnWsKKG35tI2LB9VDPiCgTy.Gq2VxQLYjrue4Nq.NBdqI- HTTP/1.1
Host: example-bucket.s3.amazonaws.com
Authorization: authorization string
Date: Wed, 28 May 2014 19:40:11 +0000
x-amz-server-side-encryption-customer-key:
g0lCfA3Dv40jZz5SQJ1ZukLRFqtI5WorC/8SEEXAMPLE
x-amz-server-side-encryption-customer-key-MD5: ZjQrne1X/iTcskbY2example
x-amz-server-side-encryption-customer-algorithm: AES256
```

In the response, Amazon S3 returns encryption-specific headers providing the encryption algorithm used and MD5 digest of the encryption key you provided in the request.

```
HTTP/1.1 100 Continue   HTTP/1.1 200 OK
x-amz-id-2: Zn8bf8aEFQ+kBnGPBc/JaAf9SoWM68QDPS9+SyFwkIZOHUG2BiRLZi5oXw4cOCEt
x-amz-request-id: 5A37448A37622243
Date: Wed, 28 May 2014 19:40:12 GMT
ETag: "7e10e7d25dc4581d89b9285be5f384fd"
x-amz-server-side-encryption-customer-algorithm: AES256
x-amz-server-side-encryption-customer-key-MD5: ZjQrne1X/iTcskbY2example
```

Related Actions

- Initiate Multipart Upload (p. 282)
- Complete Multipart Upload (p. 302)
- Abort Multipart Upload (p. 308)
- List Parts (p. 310)
- List Multipart Uploads (p. 135)

Upload Part - Copy

Description

Uploads a part by copying data from an existing object as data source. You specify the data source by adding the request header `x-amz-copy-source` in your request and a byte range by adding the request header `x-amz-copy-source-range` in your request.

The minimum allowable part size for a multipart upload is 5 MB. For more information about multipart upload limits, go to Quick Facts in the *Amazon Simple Storage Service Developer Guide*.

> **Note**
> Instead of using an existing object as part data, you might use the `Upload Part` operation and provide data in your request. For more information, see Upload Part (p. 290).

You must initiate a multipart upload before you can upload any part. In response to your initiate request. Amazon S3 returns a unique identifier, the upload ID, that you must include in your upload part request.

For more information on using the upload part - copy operation, see the following topics:

- For conceptual information on multipart uploads, go to Uploading Objects Using Multipart Upload in the *Amazon Simple Storage Service Developer Guide*.
- For information on permissions required to use the multipart upload API, go to Multipart Upload API and Permissions in the *Amazon Simple Storage Service Developer Guide*.
- For information about copying objects using a single atomic operation vs. the multipart upload, go to Operations on Objects in the *Amazon Simple Storage Service Developer Guide*.
- For information about using server-side encryption with customer-provided encryption keys with the upload part - copy operation, see PUT Object - Copy (p. 269) and Upload Part (p. 290).

Requests

Syntax

```
PUT /ObjectName?partNumber=PartNumber&uploadId=UploadId HTTP/1.1
Host: BucketName.s3.amazonaws.com
x-amz-copy-source: /source_bucket/sourceObject
x-amz-copy-source-range:bytes=first-last
x-amz-copy-source-if-match: etag
x-amz-copy-source-if-none-match: etag
x-amz-copy-source-if-unmodified-since: time_stamp
x-amz-copy-source-if-modified-since: time_stamp
Date: date
Authorization: authorization string
```

Request Parameters

This operation does not use request parameters.

Request Headers

This implementation of the operation can use the following request headers in addition to the request headers common to all operations. For more information, see Common Request Headers (p. 12).

Name	Description	Required
x-amz-copy-source	The name of the source bucket and the source object key name separated by a slash ('/'). Type: String Default: None	Yes
x-amz-copy-source-range	The range of bytes to copy from the source object. The range value must use the form `bytes=first-last`, where the first and last are the zero-based byte offsets to copy. For example, `bytes=0-9` indicates that you want to copy the first ten bytes of the source. This request header is not required when copying an entire source object. Type: Integer Default: None	No

The following conditional headers are based on the object that the *x-amz-copy-source* header specifies.

Name	Description	Required
x-amz-copy-source-if-match	Perform a copy if the source object entity tag (ETag) matches the specified value. If the value does not match, Amazon S3 returns an HTTP status code 412 *precondition failed* error. Type: String Default: None	No
x-amz-copy-source-if-none-match	Perform a copy if the source object entity tag (ETag) is different than the value specified using this header. If the values match, Amazon S3 returns an HTTP status code 412 *precondition failed* error. Type: String Default: None	No
x-amz-copy-source-if-unmodified-since	Perform a copy if the source object is not modified after the time specified using this header. If the source object is modified, Amazon S3 returns an HTTP status code 412 *precondition failed* error. Type: String Default: None	No
x-amz-copy-source-if-modified-since	Perform a copy if the source object is modified after the time specified using this header. If the source object is not modified, Amazon S3 returns an HTTP status code 412 *precondition failed* error. Type: String Default: None	No

Server-Side Encryption Specific Request Headers

If you requested server-side encryption using a customer-provided encryption key in your initiate multipart upload request, you must provide identical encryption information in each part upload using the following headers.

Name	Description	Required
x-amz-server-side-encryption-customer-algorithm	Specifies the algorithm to use to when encrypting the object. Type: String Default: None Valid Value: AES256 Constraints: Must be accompanied by a valid *x-amz-server-side-encryption-customer-key* and *x-amz-server-side-encryption-customer-key-MD5* headers.	Yes
x-amz-server-side-encryption-customer-key	Specifies the customer provided base64-encoded encryption key for Amazon S3 to use in encrypting data. This must be the same encryption key specified in the initiate multipart upload request. Type: String Default: None Constraints: Must be accompanied by a valid *x-amz-server-side-encryption-customer-algorithm* and *x-amz-server-side-encryption-customer-key-MD5* headers.	Yes
x-amz-server-side-encryption-customer-key-MD5	Specifies the base64-encoded 128-bit MD5 digest of the encryption key according to RFC 1321. Amazon S3 uses this header as a message integrity check to ensure the encryption key was transmitted without error. Type: String Default: None Constraints: Must be accompanied by a valid *x-amz-server-side-encryption-customer-algorithm* and *x-amz-server-side-encryption-customer-key* headers.	Yes

If the source object is encrypted using server-side encryption with a customer-provided encryption key, you must use the following headers providing encryption information so that Amazon S3 can decrypt the object for copying.

Name	Description	Required
x-amz-copy-source-server-side-encryption-customer-algorithm	Specifies algorithm to use when decrypting the source object. Type: String Default: None Valid Value: AES256 Constraints: Must be accompanied by a valid *x-amz-copy-source-server-side-encryption-customer-key* and *x-amz-copy-source-server-side-encryption-customer-key-MD5* headers.	Yes
x-amz-copy-source-server-side-encryption-customer-key	Specifies the customer provided base-64 encoded encryption key for Amazon S3 to use to decrypt the source object. The encryption key provided in this header must be one that was used when the source object was created. Type: String Default: None Constraints: Must be accompanied by a valid *x-amz-copy-source-server-side-encryption-customer-algorithm* and *x-amz-copy-source-server-side-encryption-customer-key-MD5* headers.	Yes
x-amz-copy-source-server-side-encryption-customer-key-MD5	Specifies the base64-encoded 128-bit MD5 digest of the encryption key according to RFC 1321. Amazon S3 uses this header for a message integrity check to ensure the encryption key was transmitted without error. Type: String Default: None Constraints: Must be accompanied by a valid *x-amz-copy-source-server-side-encryption-customer-algorithm* and *x-amz-copy-source-server-side-encryption-customer-key* headers.	Yes

Request Elements

This operation does not use request elements.

Versioning

If your bucket has versioning enabled, you could have multiple versions of the same object. By default, *x-amz-copy-source* identifies the current version of the object to copy. If the current version is a delete marker and you don't specify a versionId in the *x-amz-copy-source*, Amazon S3 returns a 404 error, because the object does not exist. If you specify versionId in the *x-amz-copy-source* and the versionId is a delete marker, Amazon S3 returns an HTTP 400 error, because you are not allowed to specify a delete marker as a version for the *x-amz-copy-source*.

You can optionally specify a specific version of the source object to copy by adding the `versionId` subresource as shown in the following example:

```
x-amz-copy-source: /bucket/object?versionId=version id
```

Responses

Response Headers

This implementation of the operation can include the following headers in addition to the response headers common to all responses. For more information, see Common Response Headers (p. 14).

Name	Description
x-amz-copy-source-version-id	The version of the source object that was copied, if you have enabled versioning on the source bucket. Type: String
x-amz-server-side-encryption	If you specified server-side encryption in your initiate multipart upload request, the response includes this header. It confirms the encryption algorithm that Amazon S3 used to encrypt the part that you uploaded. Type: String
x-amz-server-side-encryption-customer-algorithm	If server-side encryption with customer-provided encryption keys encryption was requested, the response will include this header confirming the encryption algorithm used. Type: String Valid Values: `AES256`
x-amz-server-side-encryption-customer-key-MD5	If server-side encryption with customer-provided encryption keys encryption was requested, the response includes this header to provide roundtrip message integrity verification of the customer-provided encryption key. Type: String

Response Elements

Name	Description
CopyPartResult	Container for all response elements. Type: Container Ancestor: None
ETag	Returns the ETag of the new part. Type: String Ancestor: *CopyPartResult*
LastModified	Returns the date the part was last modified. Type: String Ancestor: *CopyPartResult*

Special Errors

Error Code	Description	HTTP Status Code
`NoSuchUpload`	The specified multipart upload does not exist. The upload ID might be invalid, or the multipart upload might have been aborted or completed.	404 Not Found
`InvalidRequest`	The specified copy source is not supported as a byte-range copy source.	400 Bad Request

For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

As the following examples illustrate, when a request succeeds, Amazon S3 returns `<CopyPartResult>` in the body. If you included `versionId` in the request, Amazon S3 returns the version ID in the `x-amz-copy-source-version-id` response header.

Sample Request

The following `PUT` request uploads a part (part number 2) in a multipart upload. The request specifies a byte range from an existing object as the source of this upload. The request includes the upload ID that you get in response to your `Initiate Multipart Upload` request.

```
PUT /newobject?partNumber=2&uploadId=VCVsb2FkIElEIGZvciBlbZZpbm
cncyBteS1tb3ZpZS5tMnRzIHVwbG9hZR HTTP/1.1
Host: target-bucket.s3.amazonaws.com
Date:  Mon, 11 Apr 2011 20:34:56 GMT
x-amz-copy-source: /source-bucket/sourceobject
x-amz-copy-source-range:bytes=500-6291456
Authorization: authorization string
```

Sample Response

The response includes the ETag value. You need to retain this value to use when you send the `Complete Multipart Upload` request.

```
HTTP/1.1 200 OK
x-amz-id-2: Vvag1LuByRx9e6j5Onimru9pO4ZVKnJ2Qz7/C1NPcfTWAtRPfTaOFg==
x-amz-request-id: 656c76696e6727732072657175657374
Date:  Mon, 11 Apr 2011 20:34:56 GMT
Server: AmazonS3

<CopyPartResult>
   <LastModified>2009-10-28T22:32:00</LastModified>
   <ETag>"9b2cf535f27731c974343645a3985328"</ETag>
</CopyPartResult>
```

Sample Request

The following PUT request uploads a part (part number 2) in a multipart upload. The request does not specify the optional byte range header, but requests the entire source object copy as part 2. The request includes the upload ID that you got in response to your `Initiate Multipart Upload request`.

```
PUT /newobject?partNumber=2&uploadId=VCVsb2FkIElEIGZvciBlbZZpbm
cncyBteS1tb3ZpZS5tMnRzIHVwbG9hZR HTTP/1.1
Host: target-bucket.s3.amazonaws.com
Date:  Mon, 11 Apr 2011 20:34:56 GMT
x-amz-copy-source: /source-bucket/sourceobject
Authorization: authorization string
Sample Response
```

The response structure is similar to the one specified in the preceding example.

Sample Request

The following PUT request uploads a part (part number 2) in a multipart upload. The request specifies a specific version of the source object to copy by adding the `versionId` subresource. The byte range requests 6 MB of data, starting with byte 500, as the part to be uploaded.

```
PUT /newobject?partNumber=2&uploadId=VCVsb2FkIElEIGZvciBlbZZpbm
cncyBteS1tb3ZpZS5tMnRzIHVwbG9hZR HTTP/1.1
Host: target-bucket.s3.amazonaws.com
Date:  Mon, 11 Apr 2011 20:34:56 GMT
x-amz-copy-source: /source-bucket/sourceobject?versionId=3/L4kqtJlcpXroDTDmJ+rm
SpXd3dIbrHY+MTRCxf3vjVBH40Nr8X8gdRQBpUMLUo
x-amz-copy-source-range:bytes=500-6291456
Authorization: authorization string
```

Sample Response

The response includes the ETag value. You need to retain this value to use when you send the `Complete Multipart Upload` request.

```
HTTP/1.1 200 OK
x-amz-id-2: Vvag1LuByRx9e6j5Onimru9pO4ZVKnJ2Qz7/C1NPcfTWAtRPfTaOFg==
x-amz-request-id: 656c76696e6727732072657175657374
x-amz-copy-source-version-id: 3/L4kqtJlcpXroDTDmJ+rmSpXd3dIb
rHY+MTRCxf3vjVBH40Nr8X8gdRQBpUMLUo
Date:  Mon, 11 Apr 2011 20:34:56 GMT
Server: AmazonS3

<CopyPartResult>
   <LastModified>2009-10-28T22:32:00</LastModified>
   <ETag>"9b2cf535f27731c974343645a3985328"</ETag>
</CopyPartResult>
```

Related Actions

- Initiate Multipart Upload (p. 282)
- Upload Part (p. 290)
- Complete Multipart Upload (p. 302)
- Abort Multipart Upload (p. 308)
- List Parts (p. 310)
- List Multipart Uploads (p. 135)

Complete Multipart Upload

Description

This operation completes a multipart upload by assembling previously uploaded parts.

You first initiate the multipart upload and then upload all parts using the Upload Parts operation (see Upload Part (p. 290)). After successfully uploading all relevant parts of an upload, you call this operation to complete the upload. Upon receiving this request, Amazon S3 concatenates all the parts in ascending order by part number to create a new object. In the Complete Multipart Upload request, you must provide the parts list. You must ensure the parts list is complete, this operation concatenates the parts you provide in the list. For each part in the list, you must provide the part number and the *ETag* header value, returned after that part was uploaded.

Processing of a Complete Multipart Upload request could take several minutes to complete. After Amazon S3 begins processing the request, it sends an HTTP response header that specifies a `200 OK` response. While processing is in progress, Amazon S3 periodically sends whitespace characters to keep the connection from timing out. Because a request could fail after the initial `200 OK` response has been sent, it is important that you check the response body to determine whether the request succeeded.

Note that if Complete Multipart Upload fails, applications should be prepared to retry the failed requests. For more information, go to Amazon S3 Error Best Practices section of the *Amazon Simple Storage Service Developer Guide* .

For more information on multipart uploads, go to Uploading Objects Using Multipart Upload in the *Amazon Simple Storage Service Developer Guide* .

For information on permissions required to use the multipart upload API, go to Multipart Upload API and Permissions in the *Amazon Simple Storage Service Developer Guide* .

Requests

Syntax

```
POST /ObjectName?uploadId=UploadId HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: Date
Content-Length: Size
Authorization: authorization string

<CompleteMultipartUpload>
  <Part>
    <PartNumber>PartNumber</PartNumber>
    <ETag>ETag</ETag>
  </Part>
  ...
</CompleteMultipartUpload>
```

Request Parameters

This operation does not use request parameters.

Request Headers

This operation uses only Request Headers common to most requests. For more information, see Common Request Headers (p. 12)

Request Elements

Name	Description	Required
CompleteMultipartUpload	Container for the request. Ancestor: None Type: Container Children: One or more *Part* elements	Yes
Part	Container for elements related to a particular previously uploaded part. Ancestor: *CompleteMultipartUpload* Type: Container Children: *PartNumber*, *ETag*	Yes
PartNumber	Part number that identifies the part. Ancestor: *Part* Type: Integer	Yes
ETag	Entity tag returned when the part was uploaded. Ancestor: *Part* Type: String	Yes

Responses

Response Headers

The operation uses the following response header, in addition to the response headers common to most requests. For more information, see Common Response Headers (p. 14).

Header	Description
x-amz-expiration	Amazon S3 returns this header if an `Expiration` action is configured for the object as part of the bucket's lifecycle configuration. The header value includes an "expiry-date" component and a URL-encoded "rule-id" component. Note that for versioning-enabled buckets, this header applies only to current versions; Amazon S3 does not provide a header to infer when a noncurrent version will be eligible for permanent deletion. For more information, see PUT Bucket lifecycle (p. 162). Type: String
x-amz-server-side-encryption	If you specified server-side encryption in your initiate multipart upload request, the response includes this header confirming the encryption algorithm Amazon S3 used to save your object data to disks in its data centers. Type: String

Header	Description
x-amz-server-side-encryption-customer-algorithm	If encryption by using server-side encryption with customer-provided encryption keys was requested, the response will include this header confirming the encryption algorithm used. Type: String Valid Value: AES256
x-amz-version-id	Version ID of the newly created object, in case the bucket has versioning turned on. Type: String

Response Elements

Name	Description
CompleteMultipartUploadResult	Container for the response Type: Container Children: *Location*, *Bucket*, *Key*, *ETag* Ancestors: None
Location	The URI that identifies the newly created object. Type: URI Ancestors: *CompleteMultipartUploadResult*
Bucket	The name of the bucket that contains the newly created object. Type: String Ancestors: *CompleteMultipartUploadResult*
Key	The object key of the newly created object. Type: String Ancestors: *CompleteMultipartUploadResult*
ETag	Entity tag that identifies the newly created object's data. Objects with different object data will have different entity tags. The entity tag is an opaque string. The entity tag may or may not be an MD5 digest of the object data. If the entity tag is not an MD5 digest of the object data, it will contain one or more nonhexadecimal characters and/or will consist of less than 32 or more than 32 hexadecimal digits. Type: String Ancestors: *CompleteMultipartUploadResult*

Special Errors

Error Code	Description	HTTP Status Code
EntityTooSmall	Your proposed upload is smaller than the minimum allowed object size. Each part must be at least 5 MB in size, except the last part.	400 Bad Request

Error Code	Description	HTTP Status Code
`InvalidPart`	One or more of the specified parts could not be found. The part might not have been uploaded, or the specified entity tag might not have matched the part's entity tag.	400 Bad Request
`InvalidPartOrder`	The list of parts was not in ascending order. The parts list must be specified in order by part number.	400 Bad Request
`NoSuchUpload`	The specified multipart upload does not exist. The upload ID might be invalid, or the multipart upload might have been aborted or completed.	404 Not Found

For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

Sample Request

The following Complete Multipart Upload request specifies three parts in the *`CompleteMultipartUpload`* element.

```
POST /example-object?uploadId=AAAsb2FkIElEIGZvciBlbHZpbmcncyWeeS1tb3ZpZS5tMnRzIR
RwbG9hZA HTTP/1.1
Host: example-bucket.s3.amazonaws.com
Date:  Mon, 1 Nov 2010 20:34:56 GMT
Content-Length: 391
Authorization: authorization string

<CompleteMultipartUpload>
  <Part>
    <PartNumber>1</PartNumber>
    <ETag>"a54357aff0632cce46d942af68356b38"</ETag>
  </Part>
  <Part>
    <PartNumber>2</PartNumber>
    <ETag>"0c78aef83f66abc1fa1e8477f296d394"</ETag>
  </Part>
  <Part>
    <PartNumber>3</PartNumber>
    <ETag>"acbd18db4cc2f85cedef654fccc4a4d8"</ETag>
  </Part>
</CompleteMultipartUpload>
```

Sample Response

The following response indicates that an object was successfully assembled.

```
HTTP/1.1 200 OK
x-amz-id-2: Uuag1LuByRx9e6j5Onimru9pO4ZVKnJ2Qz7/C1NPcfTWAtRPfTaOFg==
x-amz-request-id: 656c76696e672773207265717565737 4
Date: Mon, 1 Nov 2010 20:34:56 GMT
Connection: close
Server: AmazonS3
```

```
<?xml version="1.0" encoding="UTF-8"?>
<CompleteMultipartUploadResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">

  <Location>http://Example-Bucket.s3.amazonaws.com/Example-Object</Location>
  <Bucket>Example-Bucket</Bucket>
  <Key>Example-Object</Key>
  <ETag>"3858f62230ac3c915f300c664312c11f-9"</ETag>
</CompleteMultipartUploadResult>
```

Sample Response with Error Specified in Header

The following response indicates that an error occurred before the HTTP response header was sent.

```
HTTP/1.1 403 Forbidden
x-amz-id-2: Uuag1LuByRx9e6j5Onimru9pO4ZVKnJ2Qz7/C1NPcfTWAtRPfTaOFg==
x-amz-request-id: 656c76696e672773207265717565737 4
Date:  Mon, 1 Nov 2010 20:34:56 GMT
Content-Length: 237
Connection: keep-alive
Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>AccessDenied</Code>
  <Message>Access Denied</Message>
  <RequestId>656c76696e6727732072657175657374 4</RequestId>
  <HostId>Uuag1LuByRx9e6j5Onimru9pO4ZVKnJ2Qz7/C1NPcfTWAtRPfTaOFg==</HostId>
</Error>
```

Sample Response with Error Specified in Body

The following response indicates that an error occurred after the HTTP response header was sent. Note that while the HTTP status code is `200 OK`, the request actually failed as described in the *`Error`* element.

```
HTTP/1.1 200 OK
x-amz-id-2: Uuag1LuByRx9e6j5Onimru9pO4ZVKnJ2Qz7/C1NPcfTWAtRPfTaOFg==
x-amz-request-id: 656c76696e672773207265717565737 4
Date:  Mon, 1 Nov 2010 20:34:56 GMT
Connection: close
Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>

<Error>
  <Code>InternalError</Code>
  <Message>We encountered an internal error. Please try again.</Message>
  <RequestId>656c76696e672773207265717565737 4</RequestId>
  <HostId>Uuag1LuByRx9e6j5Onimru9pO4ZVKnJ2Qz7/C1NPcfTWAtRPfTaOFg==</HostId>
</Error>
```

Related Actions

- Initiate Multipart Upload (p. 282)
- Upload Part (p. 290)

- Abort Multipart Upload (p. 308)
- List Parts (p. 310)
- List Multipart Uploads (p. 135)

Abort Multipart Upload

Description

This operation aborts a multipart upload. After a multipart upload is aborted, no additional parts can be uploaded using that upload ID. The storage consumed by any previously uploaded parts will be freed. However, if any part uploads are currently in progress, those part uploads might or might not succeed. As a result, it might be necessary to abort a given multipart upload multiple times in order to completely free all storage consumed by all parts. To verify that all parts have been removed, so you don't get charged for the part storage, you should call the List Parts (p. 310) operation and ensure the parts list is empty.

For information on permissions required to use the multipart upload API, go to Multipart Upload API and Permissions in the *Amazon Simple Storage Service Developer Guide* .

Requests

Syntax

```
DELETE /ObjectName?uploadId=UploadId HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: Date
Authorization: authorization string
```

Request Parameters

This operation does not use request parameters.

Request Headers

This operation uses only Request Headers common to most requests. For more information, see Common Request Headers (p. 12).

Request Elements

This operation does not use request elements.

Responses

Response Headers

This operation uses only response headers that are common to most responses. For more information, see Common Response Headers (p. 14).

Response Elements

This operation does not use response elements.

Special Errors

Error Code	Description	HTTP Status Code	SOAP Fault Code Prefix
`NoSuchUpload`	The specified multipart upload does not exist. The upload ID might be invalid, or the multipart upload might have been aborted or completed.	404 Not Found	Client

For general information about Amazon S3 errors and a list of error codes, see Error Responses (p. 3).

Examples

Sample Request

The following request aborts a multipart upload identified by its upload ID.

```
DELETE /example-object?uploadId=VXBsb2FkIElEIGZvciBlbHZpbmcncyBteS1tb3ZpZS5tM
nRzIHVwbG9hZ HTTP/1.1
Host: example-bucket.s3.amazonaws.com
Date:  Mon, 1 Nov 2010 20:34:56 GMT
Authorization: authorization string
```

Sample Response

```
HTTP/1.1 204 OK
x-amz-id-2: Weag1LuByRx9e6j5Onimru9pO4ZVKnJ2Qz7/C1NPcfTWAtRPfTaOFg==
x-amz-request-id: 996c76696e6727732072657175657374
Date:  Mon, 1 Nov 2010 20:34:56 GMT
Content-Length: 0
Connection: keep-alive
Server: AmazonS3
```

Related Actions

- Initiate Multipart Upload (p. 282)
- Upload Part (p. 290)
- Complete Multipart Upload (p. 302)
- List Parts (p. 310)
- List Multipart Uploads (p. 135)

List Parts

Description

This operation lists the parts that have been uploaded for a specific multipart upload.

This operation must include the upload ID, which you obtain by sending the initiate multipart upload request (see Initiate Multipart Upload (p. 282)). This request returns a maximum of 1,000 uploaded parts. The default number of parts returned is 1,000 parts. You can restrict the number of parts returned by specifying the `max-parts` request parameter. If your multipart upload consists of more than 1,000 parts, the response returns an `IsTruncated` field with the value of `true`, and a `NextPartNumberMarker` element. In subsequent List Parts requests you can include the `part-number-marker` query string parameter and set its value to the `NextPartNumberMarker` field value from the previous response.

For more information on multipart uploads, go to Uploading Objects Using Multipart Upload in the *Amazon Simple Storage Service Developer Guide* .

For information on permissions required to use the multipart upload API, go to Multipart Upload API and Permissions in the *Amazon Simple Storage Service Developer Guide* .

Requests

Syntax

```
GET /ObjectName?uploadId=UploadId HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: Date
Authorization: authorization string
```

Request Parameters

This implementation of `GET` uses the parameters in the following table to return a subset of the objects in a bucket.

Parameter	Description	Required
`encoding-type`	Requests Amazon S3 to encode the response and specifies the encoding method to use. An object key can contain any Unicode character; however, XML 1.0 parser cannot parse some characters, such as characters with an ASCII value from 0 to 10. For characters that are not supported in XML 1.0, you can add this parameter to request that Amazon S3 encode the keys in the response. Type: String Default: None Valid value: `url`	No
`uploadId`	Upload ID identifying the multipart upload whose parts are being listed. Type: String Default: None	Yes

Parameter	Description	Required
max-parts	Sets the maximum number of parts to return in the response body. Type: String Default: 1,000	No
part-number-marker	Specifies the part after which listing should begin. Only parts with higher part numbers will be listed. Type: String Default: None	No

Request Headers

This operation uses only Request Headers common to most requests. For more information, see Common Request Headers (p. 12).

Request Elements

This operation does not use request elements.

Responses

Response Headers

This operation uses only response headers that are common to most responses. For more information, see Common Response Headers (p. 14).

Response Elements

Name	Description
`ListPartsResult`	Container for the response. Children: *Bucket*, *Key*, *UploadId*, *Initiator*, *Owner*, *StorageClass*, *PartNumberMarker*, *NextPartNumberMarker*, *MaxParts*, *IsTruncated*, *Part* Type: Container
`Bucket`	Name of the bucket to which the multipart upload was initiated. Type: String Ancestor: *ListPartsResult*
`Encoding-Type`	Encoding type used by Amazon S3 to encode object key names in the XML response. If you specify `encoding-type` request parameter, Amazon S3 includes this element in the response, and returns encoded key name values in the Key element. Type: String Ancestor: ListBucketResult
`Key`	Object key for which the multipart upload was initiated. Type: String Ancestor: *ListPartsResult*

Name	Description
`UploadId`	Upload ID identifying the multipart upload whose parts are being listed. Type: String Ancestor: *`ListPartsResult`*
`Initiator`	Container element that identifies who initiated the multipart upload. If the initiator is an AWS account, this element provides the same information as the *`Owner`* element. If the initiator is an IAM User, then this element provides the user ARN and display name. Children: *`ID`*, *`DisplayName`* Type: Container Ancestor: *`ListPartsResult`*
`ID`	If the principal is an AWS account, it provides the Canonical User ID. If the principal is an IAM User, it provides a user ARN value. Type: String Ancestor: *`Initiator`*
`DisplayName`	Principal's name. Type: String Ancestor: *`Initiator`*
`Owner`	Container element that identifies the object owner, after the object is created. If multipart upload is initiated by an IAM user, this element provides the parent account ID and display name. Children: *`ID`*, *`DisplayName`* Type: Container Ancestor: *`ListPartsResult`*
`StorageClass`	Class of storage (`STANDARD` or `REDUCED_REDUNDANCY`) used to store the uploaded object. Type: String Ancestor: *`ListPartsResult`*
`PartNumberMarker`	Part number after which listing begins. Type: Integer Ancestor: *`ListPartsResult`*
`NextPartNumberMarker`	When a list is truncated, this element specifies the last part in the list, as well as the value to use for the *`part-number-marker`* request parameter in a subsequent request. Type: Integer Ancestor: *`ListPartsResult`*
`MaxParts`	Maximum number of parts that were allowed in the response. Type: Integer Ancestor: *`ListPartsResult`*

Name	Description
`IsTruncated`	Indicates whether the returned list of parts is truncated. A *`true`* value indicates that the list was truncated. A list can be truncated if the number of parts exceeds the limit returned in the *`MaxParts`* element. Type: Boolean Ancestor: *`ListPartsResult`*
`Part`	Container for elements related to a particular part. A response can contain zero or more *`Part`* elements. Children: *`PartNumber`*, *`LastModified`*, *`ETag`*, *`Size`* Type: String Ancestor: *`ListPartsResult`*
`PartNumber`	Part number identifying the part. Type: Integer Ancestor: *`Part`*
`LastModified`	Date and time at which the part was uploaded. Type: Date Ancestor: *`Part`*
`ETag`	Entity tag returned when the part was uploaded. Type: String Ancestor: *`Part`*
`Size`	Size of the uploaded part data. Type: Integer Ancestor: *`Part`*

Examples

Sample Request

Assume you have uploaded parts with sequential part numbers starting with 1. The following List Parts request specifies *`max-parts`* and *`part-number-marker`* query parameters. The request lists the first two parts that follow part number 1, that is, you will get parts 2 and 3 in the response. If more parts exist , the result is a truncated result and therefore the response will return an *`IsTruncated`* element with the value `true`. The response will also return the *`NextPartNumberMarker`* element with the value 3, which should be used for the value of the *`part-number-marker`* request query string parameter in the next List Parts request.

```
GET /example-object?uploadId=XXBsb2FkIElEIGZvciBlbHZpbmcncyVcdS1tb3ZpZS5tMnRzEEEw
bG9hZA&max-parts=2&part-number-marker=1 HTTP/1.1
Host: example-bucket.s3.amazonaws.com
Date: Mon, 1 Nov 2010 20:34:56 GMT
Authorization: authorization string
```

Sample Response

The following is a sample response.

```
HTTP/1.1 200 OK
x-amz-id-2: Uuag1LuByRx9e6j5Onimru9pO4ZVKnJ2Qz7/C1NPcfTWAtRPfTaOFg==
x-amz-request-id: 656c76696e6727732072657175657374
Date: Mon, 1 Nov 2010 20:34:56 GMT
Content-Length: 985
Connection: keep-alive
Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<ListPartsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Bucket>example-bucket</Bucket>
  <Key>example-object</Key>
  <UploadId>XXBsb2FkIElEIGZvciBlbHZpbmcncyVcdS1tb3ZpZS5tMnRzEEEwbG9hZA</UploadId>

  <Initiator>
      <ID>arn:aws:iam::111122223333:user/some-user-11116a31-17b5-4fb7-9df5-
b288870f11xx</ID>
     <DisplayName>umat-user-11116a31-17b5-4fb7-9df5-b288870f11xx</DisplayName>

  </Initiator>
  <Owner>
    <ID>75aa57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
    <DisplayName>someName</DisplayName>
  </Owner>
  <StorageClass>STANDARD</StorageClass>
  <PartNumberMarker>1</PartNumberMarker>
  <NextPartNumberMarker>3</NextPartNumberMarker>
  <MaxParts>2</MaxParts>
  <IsTruncated>true</IsTruncated>
  <Part>
    <PartNumber>2</PartNumber>
    <LastModified>2010-11-10T20:48:34.000Z</LastModified>
    <ETag>"7778aef83f66abc1fa1e8477f296d394"</ETag>
    <Size>10485760</Size>
  </Part>
  <Part>
    <PartNumber>3</PartNumber>
    <LastModified>2010-11-10T20:48:33.000Z</LastModified>
    <ETag>"aaaa18db4cc2f85cedef654fccc4a4x8"</ETag>
    <Size>10485760</Size>
  </Part>
</ListPartsResult>
```

Related Actions

- Initiate Multipart Upload (p. 282)
- Upload Part (p. 290)
- Complete Multipart Upload (p. 302)
- Abort Multipart Upload (p. 308)
- List Multipart Uploads (p. 135)

Amazon S3 Resources

Following is a table that lists related resources that you'll find useful as you work with this service.

Resource	Description
Amazon Simple Storage Service Getting Started Guide	The Getting Started Guide provides a quick tutorial of the service based on a simple use case. Examples and instructions for Java, Perl, PHP, C#, Python, and Ruby are included.
Amazon S3 Developer Guide	The developer guide describes how to accomplish tasks using Amazon S3 operations.
Amazon S3Technical FAQ	The FAQ covers the top 20 questions developers have asked about this product.
Amazon S3 Release Notes	The Release Notes give a high-level overview of the current release. They specifically note any new features, corrections, and known issues.
AWS Developer Resource Center	A central starting point to find documentation, code samples, release notes, and other information to help you build innovative applications with AWS.
AWS Management Console	The console allows you to perform most of the functions of Amazon S3without programming.
Discussion Forums	A community-based forum for developers to discuss technical questions related to Amazon Web Services.
AWS Support Center	The home page for AWS Technical Support, including access to our Developer Forums, Technical FAQs, Service Status page, and Premium Support.
AWS Premium Support	The primary web page for information about AWS Premium Support, a one-on-one, fast-response support channel to help you build and run applications on AWS Infrastructure Services.
Amazon S3 product information	The primary web page for information about Amazon S3.

Resource	Description
Contact Us	A central contact point for inquiries concerning AWS billing, account, events, abuse etc.
Conditions of Use	Detailed information about the copyright and trademark usage at Amazon.com and other topics.

Document History

The following table describes the important changes to the documentation since the last release of the *Amazon Simple Storage Service API Reference*.

- **API version:** 2006-03-01
- **Latest documentation update:** June 12, 2014

Change	Description	Release Date
Server-side encryption with customer-provided encryption keys	Amazon S3 now supports server-side encryption using customer-provided encryption keys (SSE-C). Server-side encryption enables you to request Amazon S3 to encrypt your data at rest. When using SSE-C, Amazon S3 encrypts your objects with the custom encryption keys that you provide. Since Amazon S3 performs the encryption for you, you get the benefits of using your own encryption keys without the cost of writing or executing your own encryption code. For more information about SSE-C, go to Server-Side Encryption (Using Customer-Provided Encryption Keys) in the *Amazon Simple Storage Service Developer Guide*. The following Amazon S3 REST APIs support headers related to SSE-C. • GET Object • HEAD Object • PUT Object • PUT Object - Copy • POST Object • Initiate Multipart Upload • Upload Part • Upload Part - Copy	June 12, 2014

Change	Description	Release Date
Lifecycle support for versioning	Prior to this release lifecycle configuration was supported only on nonversioned buckets. Now you can configure lifecycle on both the nonversioned and versioning-enabled buckets. For more information, go to Object Lifecycle Management in the Amazon Simple Storage Service Developer Guide. The related APIs, see PUT Bucket lifecycle (p. 162), GET Bucket lifecycle (p. 95), and DELETE Bucket lifecycle (p. 73).	May 20, 2014
Amazon S3 now supports Signature Version 4	Amazon S3 now supports Signature Verion 4 (SigV4) in all regions, the latest specification for how to sign and authenticate AWS requests. For more information, see Authenticating Requests (AWS Signature Version 4) (p. 15).	January 30, 2014
Amazon S3 list actions now support `encoding-type` request parameter	The following Amazon S3 list actions now support `encoding-type` optional request parameter. GET Bucket (List Objects) (p. 81) GET Bucket Object versions (p. 114) List Multipart Uploads (p. 135) List Parts (p. 310) An object key can contain any Unicode character; however, the XML 1.0 parser cannot parse some characters, such as characters with an ASCII value from 0 to 10. For characters that are not supported in XML 1.0, you can add this parameter to request that Amazon S3 encode the keys in the response.	November 1, 2013
SOAP Support Over HTTP Deprecated	SOAP support over HTTP is deprecated, but it is still available over HTTPS. New Amazon S3 features will not be supported for SOAP. We recommend that you use either the REST API or the AWS SDKs.	September 19, 2013
Root domain support for website hosting	Amazon S3 now supports hosting static websites at the root domain. Visitors to your website can access your site from their browser without specifying "www" in the web address (e.g., "example.com"). Many customers already host static websites on Amazon S3 that are accessible from a "www" subdomain (e.g., "www.example.com"). Previously, to support root domain access, you needed to run your own web server to proxy root domain requests from browsers to your website on Amazon S3. Running a web server to proxy requests introduces additional costs, operational burden, and another potential point of failure. Now, you can take advantage of the high availability and durability of Amazon S3 for both "www" and root domain addresses. For an example walkthrough, go to go to Example: Setting Up a Static Website Using a Custom Domain. For conceptual information, go to Hosting Static Websites on Amazon S3 in the *Amazon Simple Storage Service Developer Guide*.	December 27, 2012

Change	Description	Release Date
Support for Archiving Data to Amazon Glacier	Amazon S3 now support a storage option that enables you to utilize Amazon Glacier's low-cost storage service for data archival. To archive objects, you define archival rules identifying objects and a timeline when you want Amazon S3 to archive these objects to Amazon Glacier. You can easily set the rules on a bucket using the Amazon S3 console or programmatically using the Amazon S3 API or AWS SDKs. To support data archival rules, Amazon S3 lifecycle management API has been updated. For more information, see PUT Bucket lifecycle (p. 162). After you archive objects, you must first restore a copy before you can access the data. Amazon S3 offers an new API for you to initiate a restore. For more information, see POST Object restore (p. 247). For conceptual information, go to Object Lifecycle Management in the *Amazon Simple Storage Service Developer Guide*.	November 13, 2012
Support for Website Page Redirects	For a bucket that is configured as a website, Amazon S3 now supports redirecting a request for an object to another object in the same bucket or to an external URL. You can configure redirect by adding the `x-amz-website-redirect-location` metadata to the object. The object upload APIs PUT Object (p. 250), Initiate Multipart Upload (p. 282), and POST Object (p. 238) allow you to configure the `x-amz-website-redirect-location` object metadata. For conceptual information, go to How to Configure Website Page Redirects in the *Amazon Simple Storage Service Developer Guide*.	October 4, 2012
Cross-Origin Resource Sharing (CORS) support	Amazon S3 now supports Cross-Origin Resource Sharing (CORS). CORS defines a way in which client web applications that are loaded in one domain can interact with or access resources in a different domain. With CORS support in Amazon S3, you can build rich client-side web applications on top of Amazon S3 and selectively allow cross-domain access to your Amazon S3 resources. For more information, see Enabling Cross-Origin Resource Sharing in the *Amazon Simple Storage Service Developer Guide*.	August 31, 2012
Cost Allocation Tagging support	Amazon S3 now supports cost allocation tagging, which allows you to label S3 buckets so you can more easily track their cost against projects or other criteria. For more information, see Cost Allocation Tagging in the *Amazon Simple Storage Service Developer Guide*.	August 21, 2012
Object Expiration support	You can use Object Expiration to schedule automatic removal of data after a configured time period. You set object expiration by adding lifecycle configuration to a bucket. For more information, see Object Expiration.	December 27, 2011

Change	Description	Release Date
New Region supported	Amazon S3 now supports the South America (Sao Paulo) Region. For more information, see Buckets and Regions in the *Amazon Simple Storage Service Developer Guide*.	December 14, 2011
Multi-Object Delete	Amazon S3 now supports Multi-Object Delete API that enables you to delete multiple objects in a single request. With this feature, you can remove large numbers of objects from Amazon S3 more quickly than using multiple individual DELETE requests. For more information about the API see, see Delete Multiple Objects (p. 203). For conceptual information about the delete operation, see Deleting Objects.	December 7, 2011
New Region supported	Amazon S3 now supports the US West (Oregon) Region. For more information, see Buckets and Regions in the *Amazon Simple Storage Service Developer Guide*.	November 8, 2011
Server-side encryption support	Amazon S3 now supports server-side encryption. It enables you to request Amazon S3 to encrypt your data at rest, that is, encrypt your object data when Amazon S3 writes your data to disks in its data centers. To request server-side encryption, you must add the `x-amz-server-side-encryption` header to your request. To learn more about data encryption, go to Using Data Encryption.	October 17, 2011
Multipart Upload API extended to enable copying objects up to 5 TB	Prior to this release, Amazon S3 API supported copying objects (see PUT Object - Copy (p. 269)) of up to 5 GB in size. To enable copying objects larger than 5 GB, Amazon S3 extends the multipart upload API with a new operation, `Upload Part (Copy)`. You can use this multipart upload operation to copy objects up to 5 TB in size. For conceptual information about multipart upload, go to Uploading Objects Using Multipart Upload. To learn more about the new API, see Upload Part - Copy (p. 295).	June 21, 2011
SOAP API calls over HTTP disabled	To increase security, SOAP API calls over HTTP are disabled. Authenticated and anonymous SOAP requests must be sent to Amazon S3 using SSL.	June 6, 2011

Change	Description	Release Date
Support for hosting static websites in Amazon S3	Amazon S3 introduces enhanced support for hosting static websites. This includes support for index documents and custom error documents. When using these features, requests to the root of your bucket or a subfolder (e.g., `http://mywebsite.com/subfolder`) returns your index document instead of the list of objects in your bucket. If an error is encountered, Amazon S3 returns your custom error message instead of an Amazon S3 error message. For API information to configure your bucket as a website, see the following sections: • PUT Bucket website (p. 191) • GET Bucket website (p. 131) • DELETE Bucket website (p. 79) For conceptual overview, go to Hosting Websites on Amazon S3 in the *Amazon Simple Storage Service Developer Guide*.	February 17, 2011
Response Header API Support	The GET Object REST API now allows you to change the response headers of the REST GET Object request for each request. That is, you can alter object metadata in the response, without altering the object itself. For more information, see GET Object (p. 212).	January 14, 2011
Large Object Support	Amazon S3 has increased the maximum size of an object you can store in an S3 bucket from 5 GB to 5 TB. If you are using the REST API you can upload objects of up to 5 GB size in a single PUT operation. For larger objects, you must use the Multipart Upload REST API to upload objects in parts. For conceptual information, go to Uploading Objects Using Multipart Upload. For multipart upload API information, see Initiate Multipart Upload (p. 282), Upload Part (p. 290), Complete Multipart Upload (p. 302), List Parts (p. 310), and List Multipart Uploads (p. 135)	December 9, 2010
Multipart upload	Multipart upload enables faster, more flexible uploads into Amazon S3. It allows you to upload a single object as a set of parts. For conceptual information, go to Uploading Objects Using Multipart Upload. For multipart upload API information, see Initiate Multipart Upload (p. 282), Upload Part (p. 290), Complete Multipart Upload (p. 302), List Parts (p. 310), and List Multipart Uploads (p. 135)	November 10, 2010
Notifications	The Amazon S3 notifications feature enables you to configure a bucket so that Amazon S3 publishes a message to an Amazon Simple Notification Service (SNS) topic when Amazon S3 detects a key event on a bucket. For more information, see GET Bucket notification (p. 108) and PUT Bucket notification (p. 108).	July 14, 2010
Bucket policies	Bucket policies is an access management system you use to set access permissions on buckets, objects, and sets of objects. This functionality supplements and in many cases replaces access control lists.	July 6, 2010

Change	Description	Release Date
Reduced Redundancy	Amazon S3 now enables you to reduce your storage costs by storing objects in Amazon S3 with reduced redundancy. For more information, see PUT Object (p. 250).	May 12, 2010
New Region supported	Amazon S3 now supports the Asia Pacific (Singapore) Region and therefore new location constraints. For more information, see GET Bucket location (p. 103) and PUT Bucket (p. 144).	April 28, 2010
Object Versioning	This release introduces object Versioning. All objects now have a key and a version. If you enable versioning for a bucket, Amazon S3 gives all objects added to a bucket a unique version ID. This feature enables you to recover from unintended overwrites and deletions. For more information, see GET Object (p. 212), DELETE Object (p. 200), PUT Object (p. 250), PUT Object Copy (p. 269), or POST Object (p. 238). The SOAP API does not support versioned objects.	February 8, 2010
New Region supported	Amazon S3 now supports the US-West (Northern California) Region. The new endpoint is `s3-us-west-1.amazonaws.com`. For more information, see How to Select a Region for Your Buckets.	December 2, 2009
C# Library Support	AWS now provides Amazon S3 C# libraries, sample code, tutorials, and other resources for software developers who prefer to build applications using language-specific APIs instead of REST or SOAP. These libraries provide basic functions (not included in the REST or SOAP APIs), such as request authentication, request retries, and error handling so that it's easier to get started.	November 11, 2009
Technical documents reorganized	The API reference has been split out of the *Amazon S3 Developer Guide*. Now, on the documentation landing page, http://developer.amazonwebservices.com/connect/entry.jspa?externalID=123&categoryID=48 you can select the document you want to view. When viewing the documents online, the links in one document will take you, when appropriate, to one of the other guides.	September 16, 2009

Appendix: SOAP API

> **Note**
> SOAP support over HTTP is deprecated, but it is still available over HTTPS. New Amazon S3 features will not be supported for SOAP. We recommend that you use either the REST API or the AWS SDKs.

This section describes the SOAP API with respect to service, bucket, and object operations. Note that SOAP requests, both authenticated and anonymous, must be sent to Amazon S3 using SSL. Amazon S3 returns an error when you send a SOAP request over HTTP.

Topics

- Operations on the Service (p. 323)
- Operations on Buckets (p. 325)
- Operations on Objects (p. 334)
- SOAP Error Responses (p. 350)

Operations on the Service

> **Note**
> SOAP support over HTTP is deprecated, but it is still available over HTTPS. New Amazon S3 features will not be supported for SOAP. We recommend that you use either the REST API or the AWS SDKs.

This section describes operations you can perform on the Amazon S3 service.

Topics

- ListAllMyBuckets (p. 323)

ListAllMyBuckets

> **Note**
> SOAP support over HTTP is deprecated, but it is still available over HTTPS. New Amazon S3 features will not be supported for SOAP. We recommend that you use either the REST API or the AWS SDKs.

The `ListAllMyBuckets` operation returns a list of all buckets owned by the sender of the request.

Example

Sample Request

```
<ListAllMyBuckets xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</ListAllMyBuckets>
```

Sample Response

```
<ListAllMyBucketsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01">
  <Owner>
    <ID>bcaf1ffd86f41161ca5fb16fd081034f</ID>
    <DisplayName>webfile</DisplayName>
  </Owner>
  <Buckets>
    <Bucket>
      <Name>quotes;/Name>
      <CreationDate>2006-02-03T16:45:09.000Z</CreationDate>
    </Bucket>
    <Bucket>
      <Name>samples</Name>
      <CreationDate>2006-02-03T16:41:58.000Z</CreationDate>
    </Bucket>
 </Buckets>
</ListAllMyBucketsResult>
```

Response Body

- *Owner:*

  This provides information that Amazon S3 uses to represent your identity for purposes of authentication and access control. ID is a unique and permanent identifier for the developer who made the request. DisplayName is a human-readable name representing the developer who made the request. It is not unique, and might change over time.We recommend that you match your DisplayName to your Forum name.

- *Name:*

  The name of a bucket. Note that if one of your buckets was recently deleted, the name of the deleted bucket might still be present in this list for a period of time.

- *CreationDate:*

  The time that the bucket was created.

Access Control

You must authenticate with a valid AWS Access Key ID. Anonymous requests are never allowed to list buckets, and you can only list buckets for which you are the owner.

Operations on Buckets

Note
SOAP support over HTTP is deprecated, but it is still available over HTTPS. New Amazon S3 features will not be supported for SOAP. We recommend that you use either the REST API or the AWS SDKs.

This section describes operations you can perform on Amazon S3 buckets.

Topics

- CreateBucket (p. 325)
- DeleteBucket (p. 326)
- ListBucket (p. 327)
- GetBucketAccessControlPolicy (p. 330)
- SetBucketAccessControlPolicy (p. 331)
- GetBucketLoggingStatus (p. 332)
- SetBucketLoggingStatus (p. 333)

CreateBucket

Note
SOAP support over HTTP is deprecated, but it is still available over HTTPS. New Amazon S3 features will not be supported for SOAP. We recommend that you use either the REST API or the AWS SDKs.

The `CreateBucket` operation creates a bucket. Not every string is an acceptable bucket name. For information on bucket naming restrictions, see Working with Amazon S3 Buckets .

Note
To determine whether a bucket name exists, use `ListBucket` and set `MaxKeys` to 0. A NoSuchBucket response indicates that the bucket is available, an AccessDenied response indicates that someone else owns the bucket, and a Success response indicates that you own the bucket or have permission to access it.

Example Create a bucket named "quotes"

Sample Request

```
<CreateBucket xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</CreateBucket>
```

Sample Response

```
<CreateBucketResponse xmlns="http://s3.amazonaws.com/doc/2006-03-01">
  <CreateBucketResponse>
    <Bucket>quotes</Bucket>
  </CreateBucketResponse>
</CreateBucketResponse>
```

Elements

- *Bucket:* The name of the bucket you are trying to create.
- *AccessControlList:* The access control list for the new bucket. This element is optional. If not provided, the bucket is created with an access policy that give the requester FULL_CONTROL access.

Access Control

You must authenticate with a valid AWS Access Key ID. Anonymous requests are never allowed to create buckets.

Related Resources

- ListBucket (p. 327)

DeleteBucket

Note
SOAP support over HTTP is deprecated, but it is still available over HTTPS. New Amazon S3 features will not be supported for SOAP. We recommend that you use either the REST API or the AWS SDKs.

The `DeleteBucket` operation deletes a bucket. All objects in the bucket must be deleted before the bucket itself can be deleted.

Example

This example deletes the "quotes" bucket.

Sample Request

```
<DeleteBucket xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <AWSAccessKeyId> AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</DeleteBucket>
```

Sample Response

```
<DeleteBucketResponse xmlns="http://s3.amazonaws.com/doc/2006-03-01">
  <DeleteBucketResponse>
    <Code>204</Code>
    <Description>No Content</Description>
  </DeleteBucketResponse>
</DeleteBucketResponse>
```

Elements

- *Bucket:* The name of the bucket you want to delete.

Access Control

Only the owner of a bucket is allowed to delete it, regardless the access control policy on the bucket.

ListBucket

> **Note**
> SOAP support over HTTP is deprecated, but it is still available over HTTPS. New Amazon S3 features will not be supported for SOAP. We recommend that you use either the REST API or the AWS SDKs.

The `ListBucket` operation returns information about some of the items in the bucket.

For a general introduction to the list operation, see the Listing Object Keys.

Requests

This example lists up to 1000 keys in the "quotes" bucket that have the prefix "notes."

Syntax

```
<ListBucket xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Prefix>notes/</Prefix>
  <Delimiter>/</Delimiter>
  <MaxKeys>1000</MaxKeys>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</ListBucket>
```

Parameters

Name	Description	Required
prefix	Limits the response to keys which begin with the indicated prefix. You can use prefixes to separate a bucket into different sets of keys in a way similar to how a file system uses folders. Type: String Default: None	No
marker	Indicates where in the bucket to begin listing. The list will only include keys that occur lexicographically after marker. This is convenient for pagination: To get the next page of results use the last key of the current page as the marker. Type: String Default: None	No
max-keys	The maximum number of keys you'd like to see in the response body. The server might return fewer than this many keys, but will not return more. Type: String Default: None	No

Name	Description	Required
delimiter	Causes keys that contain the same string between the prefix and the first occurrence of the delimiter to be rolled up into a single result element in the CommonPrefixes collection. These rolled-up keys are not returned elsewhere in the response. Type: String Default: None	No

Success Response

This response assumes the bucket contains the following keys:

```
notes/todos.txt
notes/2005-05-23/customer_mtg_notes.txt
notes/2005-05-23/phone_notes.txt
notes/2005-05-28/sales_notes.txt
```

Syntax

```
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>backups</Name>
  <Prefix>notes/</Prefix>
  <MaxKeys>1000</MaxKeys>
  <Delimiter>/</Delimiter>
  <IsTruncated>false</IsTruncated>
  <Contents>
    <Key>notes/todos.txt</Key>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>&quot;828ef3fdfa96f00ad9f27c383fc9ac7f&quot;</ETag>
    <Size>5126</Size>
    <StorageClass>STANDARD</StorageClass>
    <Owner>
     <ID>75aa57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>

      <DisplayName>webfile</DisplayName>
    </Owner>
    <StorageClass>STANDARD</StorageClass>
  </Contents>
  <CommonPrefixes>
    <Prefix>notes/2005-05-23/</Prefix>
  </CommonPrefixes>
  <CommonPrefixes>
    <Prefix>notes/2005-05-28/</Prefix>
  </CommonPrefixes>
  </ListBucketResult>
```

As you can see, many of the fields in the response echo the request parameters. *IsTruncated*, *Contents*, and *CommonPrefixes* are the only response elements that can contain new information.

Response Elements

Name	Description
Contents	Metadata about each object returned. Type: XML metadata Ancestor: ListBucketResult
CommonPrefixes	A response can contain *CommonPrefixes* only if you specify a *delimiter*. When you do, *CommonPrefixes* contains all (if there are any) keys between *Prefix* and the next occurrence of the string specified by *delimiter*. In effect, *CommonPrefixes* lists keys that act like subdirectories in the directory specified by *Prefix*. For example, if *prefix* is notes/ and *delimiter* is a slash (/), in notes/summer/july, the common prefix is *notes/summer/*. Type: String Ancestor: ListBucketResult
Delimiter	Causes keys that contain the same string between the prefix and the first occurrence of the delimiter to be rolled up into a single result element in the CommonPrefixes collection. These rolled-up keys are not returned elsewhere in the response. Type: String Ancestor: ListBucketResult
IsTruncated	Specifies whether (true) or not (false) all of the results were returned. All of the results may not be returned if the number of results exceeds that specified by *MaxKeys*. Type: String Ancestor: boolean
Marker	Indicates where in the bucket to begin listing. Type: String Ancestor: ListBucketResult
MaxKeys	The maximum number of keys returned in the response body. Type: String Ancestor: ListBucketResult
Name	Name of the bucket. Type: String Ancestor: ListBucketResult
Prefix	Keys that begin with the indicated prefix. Type: String Ancestor: ListBucketResult

Response Body

For information about the list response, see Listing Keys Response.

Access Control

To list the keys of a bucket you need to have been granted `READ` access on the bucket.

GetBucketAccessControlPolicy

> **Note**
> SOAP support over HTTP is deprecated, but it is still available over HTTPS. New Amazon S3 features will not be supported for SOAP. We recommend that you use either the REST API or the AWS SDKs.

The `GetBucketAccessControlPolicy` operation fetches the access control policy for a bucket.

Example

This example retrieves the access control policy for the "quotes" bucket.

Sample Request

```
<GetBucketAccessControlPolicy xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetBucketAccessControlPolicy>
```

Sample Response

```
<AccessControlPolicy>
  <Owner>
    <ID>a9a7b886d6fd2441bf9b1c61be666e9</ID>
    <DisplayName>chriscustomer</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>a9a7b886d6f41bf9b1c61be666e9</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/global/AllUsers<URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
<AccessControlPolicy>
```

Response Body

The response contains the access control policy for the bucket. For an explanation of this response, see SOAP Access Policy .

Access Control

You must have `READ_ACP` rights to the bucket in order to retrieve the access control policy for a bucket.

SetBucketAccessControlPolicy

> **Note**
> SOAP support over HTTP is deprecated, but it is still available over HTTPS. New Amazon S3 features will not be supported for SOAP. We recommend that you use either the REST API or the AWS SDKs.

The `SetBucketAccessControlPolicy` operation sets the Access Control Policy for an existing bucket. If successful, the previous Access Control Policy for the bucket is entirely replaced with the specified Access Control Policy.

Example

Give the specified user (usually the owner) `FULL_CONTROL` access to the "quotes" bucket.

Sample Request

```
<SetBucketAccessControlPolicy xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>a9a7b8863000e241bf9b1c61be666e9</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</SetBucketAccessControlPolicy >
```

Sample Response

```
<GetBucketAccessControlPolicyResponse xmlns="http://s3.amazonaws.com/doc/2006-
03-01">
  <GetBucketAccessControlPolicyResponse>
    <Code>200</Code>
    <Description>OK</Description>
  </GetBucketAccessControlPolicyResponse>
</GetBucketAccessControlPolicyResponse>
```

Access Control

You must have `WRITE_ACP` rights to the bucket in order to set the access control policy for a bucket.

GetBucketLoggingStatus

Note
SOAP support over HTTP is deprecated, but it is still available over HTTPS. New Amazon S3 features will not be supported for SOAP. We recommend that you use either the REST API or the AWS SDKs.

The `GetBucketLoggingStatus` retrieves the logging status for an existing bucket.

For a general introduction to this feature, see Server Logs.

Example

Sample Request

```
<?xml version="1.0" encoding="utf-8"?>
    <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xm
lns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="ht
tp://www.w3.org/2001/XMLSchema">
      <soap:Body>
       <GetBucketLoggingStatus xmlns="http://doc.s3.amazonaws.com/2006-03-01">

          <Bucket>mybucket</Bucket>
          <AWSAccessKeyId>YOUR_AWS_ACCESS_KEY_ID</AWSAccessKeyId>
          <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
          <Signature>YOUR_SIGNATURE_HERE</Signature>
        </GetBucketLoggingStatus>
      </soap:Body>
    </soap:Envelope>
```

Sample Response

```
<?xml version="1.0" encoding="utf-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="ht
tp://www.w3.org/2001/XMLSchema-instance" >
      <soapenv:Header>
      </soapenv:Header>
      <soapenv:Body>
       <GetBucketLoggingStatusResponse xmlns="http://s3.amazonaws.com/doc/2006-
03-01">
          <GetBucketLoggingStatusResponse>
            <LoggingEnabled>
              <TargetBucket>mylogs</TargetBucket>
              <TargetPrefix>mybucket-access_log-</TargetPrefix>
            </LoggingEnabled>
          </GetBucketLoggingStatusResponse>
        </GetBucketLoggingStatusResponse>
      </soapenv:Body>
    </soapenv:Envelope>
```

Access Control

Only the owner of a bucket is permitted to invoke this operation.

SetBucketLoggingStatus

Note
SOAP support over HTTP is deprecated, but it is still available over HTTPS. New Amazon S3 features will not be supported for SOAP. We recommend that you use either the REST API or the AWS SDKs.

The `SetBucketLoggingStatus` operation updates the logging status for an existing bucket.

For a general introduction to this feature, see Server Logs.

Example

This sample request enables server access logging for the 'mybucket' bucket, and configures the logs to be delivered to 'mylogs' under prefix 'access_log-'

Sample Request

```
<?xml version="1.0" encoding="utf-8"?>
    <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xm
lns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="ht
tp://www.w3.org/2001/XMLSchema">
    <soap:Body>
    <SetBucketLoggingStatus xmlns="http://doc.s3.amazonaws.com/2006-03-01">
      <Bucket>myBucket</Bucket>
      <AWSAccessKeyId>YOUR_AWS_ACCESS_KEY_ID</AWSAccessKeyId>
      <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
      <Signature>YOUR_SIGNATURE_HERE</Signature>
      <BucketLoggingStatus>
        <LoggingEnabled>
          <TargetBucket>mylogs</TargetBucket>
          <TargetPrefix>mybucket-access_log-</TargetPrefix>
        </LoggingEnabled>
      </BucketLoggingStatus>
    </SetBucketLoggingStatus>
    </soap:Body>
    :</soap:Envelope>
```

Sample Response

```
<?xml version="1.0" encoding="utf-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="ht
tp://www.w3.org/2001/XMLSchema-instance" >
      <soapenv:Header>
      </soapenv:Header>
      <soapenv:Body>
       <SetBucketLoggingStatusResponse xmlns="http://s3.amazonaws.com/doc/2006-
03-01"/>
      </soapenv:Body>
    </soapenv:Envelope>
```

Access Control

Only the owner of a bucket is permitted to invoke this operation.

Operations on Objects

> **Note**
> SOAP support over HTTP is deprecated, but it is still available over HTTPS. New Amazon S3 features will not be supported for SOAP. We recommend that you use either the REST API or the AWS SDKs.

This section describes operations you can perform on Amazon S3 objects.

Topics

- PutObjectInline (p. 334)
- PutObject (p. 336)
- CopyObject (p. 338)
- GetObject (p. 342)
- GetObjectExtended (p. 347)
- DeleteObject (p. 348)
- GetObjectAccessControlPolicy (p. 348)
- SetObjectAccessControlPolicy (p. 349)

PutObjectInline

> **Note**
> SOAP support over HTTP is deprecated, but it is still available over HTTPS. New Amazon S3 features will not be supported for SOAP. We recommend that you use either the REST API or the AWS SDKs.

The `PutObjectInline` operation adds an object to a bucket. The data for the object is provided in the body of the SOAP message.

If an object already exists in a bucket, the new object will overwrite it because Amazon S3 stores the last write request. However, Amazon S3 is a distributed system. If Amazon S3 receives multiple write requests for the same object nearly simultaneously, all of the objects might be stored, even though only one wins in the end. Amazon S3 does not provide object locking; if you need this, make sure to build it into your application layer.

To ensure an object is not corrupted over the network, you can calculate the MD5 of an object, PUT it to Amazon S3, and compare the returned Etag to the calculated MD5 value.

PutObjectInline is not suitable for use with large objects. The system limits this operation to working with objects 1MB or smaller. PutObjectInline will fail with the *`InlineDataTooLargeError`* status code if the Data parameter encodes an object larger than 1MB. To upload large objects, consider using the non-inline PutObject API, or the REST API instead.

Example

This example writes some text and metadata into the "Nelson" object in the "quotes" bucket, give a user (usually the owner) `FULL_CONTROL` access to the object, and make the object readable by anonymous parties.

Sample Request

```
<PutObjectInline xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <Metadata>
    <Name>Content-Type</Name>
    <Value>text/plain</Value>
  </Metadata>
  <Metadata>
    <Name>family</Name>
    <Value>Muntz</Value>
  </Metadata>
  <Data>aGEtaGE=</Data>
  <ContentLength>5</ContentLength>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>a9a7b886d6fde241bf9b1c61be666e9</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</PutObjectInline>
```

Sample Response

```
<PutObjectInlineResponse xmlns="http://s3.amazonaws.com/doc/2006-03-01">
  <PutObjectInlineResponse>
    <ETag>&quot828ef3fdfa96f00ad9f27c383fc9ac7f&quot</ETag>
    <LastModified>2006-01-01T12:00:00.000Z</lastModified>
  </PutObjectInlineResponse>
</PutObjectInlineResponse>
```

Elements

- *Bucket:* The bucket in which to add the object.
- *Key:* The key to assign to the object.

- *Metadata:* You can provide name-value metadata pairs in the metadata element. These will be stored with the object.
- *Data:* The base 64 encoded form of the data.
- *ContentLength:* The length of the data in bytes.
- *AccessControlList:* An Access Control List for the resource. This element is optional. If omitted, the requester is given FULL_CONTROL access to the object. If the object already exists, the preexisting access control policy is replaced.

Responses

- *ETag:* The entity tag is an MD5 hash of the object that you can use to do conditional fetches of the object using GetObjectExtended. The ETag only reflects changes to the contents of an object, not its metadata.
- *LastModified:* The Amazon S3 timestamp for the saved object.

Access Control

You must have WRITE access to the bucket in order to put objects into the bucket.

Related Resources

- PutObject (p. 336)
- CopyObject (p. 338)

PutObject

Note
SOAP support over HTTP is deprecated, but it is still available over HTTPS. New Amazon S3 features will not be supported for SOAP. We recommend that you use either the REST API or the AWS SDKs.

The PutObject operation adds an object to a bucket. The data for the object is attached as a DIME attachment.

To ensure an object is not corrupted over the network, you can calculate the MD5 of an object, PUT it to Amazon S3, and compare the returned Etag to the calculated MD5 value.

If an object already exists in a bucket, the new object will overwrite it because Amazon S3 stores the last write request. However, Amazon S3 is a distributed system. If Amazon S3 receives multiple write requests for the same object nearly simultaneously, all of the objects might be stored, even though only one wins in the end. Amazon S3 does not provide object locking; if you need this, make sure to build it into your application layer.

Example

This example puts some data and metadata in the "Nelson" object of the "quotes" bucket, give a user (usually the owner) `FULL_CONTROL` access to the object, and make the object readable by anonymous parties. In this sample, the actual attachment is not shown.

Sample Request

```
<PutObject xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <Metadata>
    <Name>Content-Type</Name>
    <Value>text/plain</Value>
  </Metadata>
  <Metadata>
    <Name>family</Name>
    <Value>Muntz</Value>
  </Metadata>
  <ContentLength>5</ContentLength>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>a9a7b886d6241bf9b1c61be666e9</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/global/AllUsers<URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2007-05-11T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</PutObject>
```

Sample Response

```
<PutObjectResponse xmlns="http://s3.amazonaws.com/doc/2006-03-01">
  <PutObjectResponse>
    <ETag>&amp;quot;828ef3fdfa96f00ad9f27c383fc9ac7f&amp;quot;</ETag>
    <LastModified>2006-03-01T12:00:00.183Z</LastModified>
  </PutObjectResponse>
</PutObjectResponse>
```

Elements

- *`Bucket:`* The bucket in which to add the object.
- *`Key:`* The key to assign to the object.
- *`Metadata:`* You can provide name-value metadata pairs in the metadata element. These will be stored with the object.
- *`ContentLength:`* The length of the data in bytes.

- *AccessControlList:* An Access Control List for the resource. This element is optional. If omitted, the requester is given FULL_CONTROL access to the object. If the object already exists, the preexisting Access Control Policy is replaced.

Responses

- *ETag:* The entity tag is an MD5 hash of the object that you can use to do conditional fetches of the object using GetObjectExtended. The ETag only reflects changes to the contents of an object, not its metadata.
- *LastModified:* The Amazon S3 timestamp for the saved object.

Access Control

To put objects into a bucket, you must have WRITE access to the bucket.

Related Resources

- CopyObject (p. 338)

CopyObject

Note
SOAP support over HTTP is deprecated, but it is still available over HTTPS. New Amazon S3 features will not be supported for SOAP. We recommend that you use either the REST API or the AWS SDKs.

Description

The CopyObject operation creates a copy of an object when you specify the key and bucket of a source object and the key and bucket of a target destination.

When copying an object, you can preserve all metadata (default) or specify new metadata. However, the ACL is not preserved and is set to private for the user making the request. To override the default ACL setting, specify a new ACL when generating a copy request. For more information, see Using ACLs.

All copy requests must be authenticated. Additionally, you must have *read* access to the source object and *write* access to the destination bucket. For more information, see Using Auth Access.

To only copy an object under certain conditions, such as whether the Etag matches or whether the object was modified before or after a specified date, use the request parameters CopySourceIfUnmodifiedSince, CopyIfUnmodifiedSince, CopySourceIfMatch, or CopySourceIfNoneMatch.

Note
You might need to configure the SOAP stack socket timeout for copying large objects.

Request Syntax

```
<CopyObject xmlns="http://bucket_name.s3.amazonaws.com/2006-03-01">
  <SourceBucket>source_bucket</SourceBucket>
  <SourceObject>source_object</SourceObject>
  <DestinationBucket>destination_bucket</DestinationBucket>
  <DestinationObject>destination_object</DestinationObject>
```

```
  <MetadataDirective>{REPLACE | COPY}</MetadataDirective>
  <Metadata>
    <Name>metadata_name</Name>
    <Value>metadata_value</Value>
  </Metadata>
  ...
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="user_type">
        <ID>user_id</ID>
        <DisplayName>display_name</DisplayName>
      </Grantee>
      <Permission>permission</Permission>
    </Grant>
    ...
  </AccessControlList>
  <CopySourceIfMatch>etag</CopySourceIfMatch>
  <CopySourceIfNoneMatch>etag</CopySourceIfNoneMatch>
  <CopySourceIfModifiedSince>date_time</CopySourceIfModifiedSince>
  <CopySourceIfUnmodifiedSince>date_time</CopySourceIfUnmodifiedSince>
  <AWSAccessKeyId>AWSAccessKeyId</AWSAccessKeyId>
  <Timestamp>TimeStamp</Timestamp>
  <Signature>Signature</Signature>
</CopyObject>
```

Request Parameters

Name	Description	Required
SourceBucket	The name of the source bucket. Type: String Default: None Constraints: A valid source bucket.	Yes
SourceKey	The key name of the source object. Type: String Default: None Constraints: The key for a valid source object to which you have READ access.	Yes
DestinationBucket	The name of the destination bucket. Type: String Default: None Constraints: You must have WRITE access to the destination bucket.	Yes
DestinationKey	The key of the destination object. Type: String Default: None Constraints: You must have WRITE access to the destination bucket.	Yes

Name	Description	Required
MetadataDirective	Specifies whether the metadata is copied from the source object or replaced with metadata provided in the request. Type: String Default: COPY Valid values: COPY | REPLACE Constraints: Values other than COPY or REPLACE will result in an immediate error. You cannot copy an object to itself unless the MetadataDirective header is specified and its value set to REPLACE.	No
Metadata	Specifies metadata name-value pairs to set for the object.If MetadataDirective is set to COPY, all metadata is ignored. Type: String Default: None Constraints: None.	No
AccessControlList	Grants access to users by e-mail addresses or canonical user ID. Type: String Default: None Constraints: None	No
CopySourceIfMatch	Copies the object if its entity tag (ETag) matches the specified tag; otherwise return a PreconditionFailed. Type: String Default: None Constraints: None. If the Etag does not match, the object is not copied.	No
CopySourceIfNoneMatch	Copies the object if its entity tag (ETag) is different than the specified Etag; otherwise returns an error. Type: String Default: None Constraints: None.	No
CopySourceIfUnmodifiedSince	Copies the object if it hasn't been modified since the specified time; otherwise returns a PreconditionFailed. Type: dateTime Default: None	No
CopySourceIfModifiedSince	Copies the object if it has been modified since the specified time; otherwise returns an error. Type: dateTime Default: None	No

Response Syntax

```
<CopyObjectResponse xmlns="http://bucket_name.s3.amazonaws.com/2006-03-01">
  <CopyObjectResponse>
    <ETag>"etag"</ETag>
    <LastModified>timestamp</LastModified>
  </CopyObjectResponse>
</CopyObjectResponse>
```

Response Elements

Following is a list of response elements.

Note
The SOAP API does not return extra whitespace. Extra whitespace is only returned by the REST API.

Name	Description
Etag	Returns the etag of the new object. The ETag only reflects changes to the contents of an object, not its metadata. Type: String Ancestor: CopyObjectResult
LastModified	Returns the date the object was last modified. Type: String Ancestor: CopyObjectResult

For information about general response elements, see Using REST Error Response Headers.

Special Errors

There are no special errors for this operation. For information about general Amazon S3 errors, see List of Error Codes (p. 3).

Examples

This example copies the `flotsam` object from the `pacific` bucket to the `jetsam` object of the `atlantic` bucket, preserving its metadata.

Sample Request

```
<CopyObject xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <SourceBucket>pacific</SourceBucket>
  <SourceObject>flotsam</SourceObject>
  <DestinationBucket>atlantic</DestinationBucket>
  <DestinationObject>jetsam</DestinationObject>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2008-02-18T13:54:10.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbq7RrtSFmw=</Signature>
</CopyObject>
```

Sample Response

```
<CopyObjectResponse xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <CopyObjectResponse>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2008-02-18T13:54:10.183Z</LastModified>
  </CopyObjectResponse>
</CopyObjectResponse>
```

This example copies the "tweedledee" object from the wonderland bucket to the "tweedledum" object of the wonderland bucket, replacing its metadata.

Sample Request

```
<CopyObject xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <SourceBucket>wonderland</SourceBucket>
  <SourceObject>tweedledee</SourceObject>
  <DestinationBucket>wonderland</DestinationBucket>
  <DestinationObject>tweedledum</DestinationObject>
  <MetadataDirective >REPLACE</MetadataDirective >
  <Metadata>
    <Name>Content-Type</Name>
    <Value>text/plain</Value>
  </Metadata>
  <Metadata>
    <Name>relationship</Name>
    <Value>twins</Value>
  </Metadata>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2008-02-18T13:54:10.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbq7RrtSFmw=</Signature>
</CopyObject>
```

Sample Response

```
<CopyObjectResponse xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <CopyObjectResponse>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2008-02-18T13:54:10.183Z</LastModified>
  </CopyObjectResponse>
</CopyObjectResponse>
```

Related Resources

- PutObject (p. 336)
- PutObjectInline (p. 334)

GetObject

Note
SOAP support over HTTP is deprecated, but it is still available over HTTPS. New Amazon S3 features will not be supported for SOAP. We recommend that you use either the REST API or the AWS SDKs.

The `GetObject` operation returns the current version of an object. If you try to `GetObject` an object that has a delete marker as its current version, S3 returns a 404 error. You cannot use the SOAP API to retrieve a specified version of an object. To do that, use the REST API. For more information, see Versioning. For more options, use the GetObjectExtended (p. 347) operation.

Example

This example gets the "Nelson" object from the "quotes" bucket.

Sample Request

```
<GetObject xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <GetMetadata>true</GetMetadata>
  <GetData>true</GetData>
  <InlineData>true</InlineData>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetObject>
```

Sample Response

```
<GetObjectResponse xmlns="http://s3.amazonaws.com/doc/2006-03-01">
  <GetObjectResponse>
    <Status>
      <Code>200</Code>
      <Description>OK</Description>
    </Status>
    <Metadata>
      <Name>Content-Type</Name>
      <Value>text/plain</Value>
    </Metadata>
    <Metadata>
      <Name>family</Name>
      <Value>Muntz</Value>
    </Metadata>
    <Data>aGEtaGE=</Data>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>&quot;828ef3fdfa96f00ad9f27c383fc9ac7f&quot;</ETag>
  </GetObjectResponse>
</GetObjectResponse>
```

Elements

- *`Bucket:`* The bucket from which to retrieve the object.
- *`Key:`* The key that identifies the object.
- *`GetMetadata:`* The metadata is returned with the object if this is true.
- *`GetData:`* The object data is returned if this is true.
- *`InlineData:`* If this is true, then the data is returned, base 64-encoded, as part of the SOAP body of the response. If false, then the data is returned as a SOAP attachment. The InlineData option is not suitable for use with large objects. The system limits this operation to working with 1MB of data or less. A GetObject request with the InlineData flag set will fail with the *`InlineDataTooLargeError`* status

code if the resulting Data parameter would have encoded more than 1MB. To download large objects, consider calling GetObject without setting the InlineData flag, or use the REST API instead.

Returned Elements

- *Metadata:* The name-value paired metadata stored with the object.
- *Data:* If InlineData was true in the request, this contains the base 64 encoded object data.
- *LastModified:* The time that the object was stored in Amazon S3.
- *ETag:* The object's entity tag. This is a hash of the object that can be used to do conditional gets. The ETag only reflects changes to the contents of an object, not its metadata.

Access Control

You can read an object only if you have been granted READ access to the object.

SOAP Chunked and Resumable Downloads

To provide GET flexibility, Amazon S3 supports chunked and resumable downloads.

Select from the following:

- For large object downloads, you might want to break them into smaller chunks. For more information, see Range GETs (p. 344)
- For GET operations that fail, you can design your application to download the remainder instead of the entire file. For more information, see REST GET Error Recovery (p. 347)

Range GETs

For some clients, you might want to break large downloads into smaller downloads. To break a GET into smaller units, use Range.

Before you can break a GET into smaller units, you must determine its size. For example, the following request gets the size of the bigfile object.

```
<ListBucket xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>bigbucket</Bucket>
  <Prefix>bigfile</Prefix>
  <MaxKeys>1</MaxKeys>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</ListBucket>
```

Amazon S3 returns the following response.

```
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01">
  <Name>quotes</Name>
  <Prefix>N</Prefix>
  <MaxKeys>1</MaxKeys>
  <IsTruncated>false</IsTruncated>
  <Contents>
    <Key>bigfile</Key>
```

```
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>&amp;quot;828ef3fdfa96f00ad9f27c383fc9ac7f&amp;quot;</ETag>
    <Size>2023276</Size>
    <StorageClass>STANDARD</StorageClass>
    <Owner>
      <ID>bcaf1ffd86f41161ca5fb16fd081034f</ID>
      <DisplayName>bigfile</DisplayName>
     </Owner>
  </Contents>
</ListBucketResult>
```

Following is a request that downloads the first megabyte from the bigfile object.

```
<GetObject xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>bigbucket</Bucket>
  <Key>bigfile</Key>
  <GetMetadata>true</GetMetadata>
  <GetData>true</GetData>
  <InlineData>true</InlineData>
  <ByteRangeStart>0</ByteRangeStart>
  <ByteRangeEnd>1048576</ByteRangeEnd>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetObject>
```

Amazon S3 returns the first megabyte of the file and the Etag of the file.

```
<GetObjectResponse xmlns="http://s3.amazonaws.com/doc/2006-03-01">
  <GetObjectResponse>
    <Status>
      <Code>200</Code>
      <Description>OK</Description>
    </Status>
    <Metadata>
      <Name>Content-Type</Name>
      <Value>text/plain</Value>
    </Metadata>
    <Metadata>
      <Name>family</Name>
      <Value>Muntz</Value>
    </Metadata>
    <Data>--first megabyte of bigfile--</Data>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
  </GetObjectResponse>
</GetObjectResponse>
```

To ensure the file did not change since the previous portion was downloaded, specify the IfMatch element. Although the IfMatch element is not required, it is recommended for content that is likely to change.

The following is a request that gets the remainder of the file, using the IfMatch request header.

```
<GetObject xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>bigbucket</Bucket>
```

```
  <Key>bigfile</Key>
  <GetMetadata>true</GetMetadata>
  <GetData>true</GetData>
  <InlineData>true</InlineData>
  <ByteRangeStart>10485761</ByteRangeStart>
  <ByteRangeEnd>2023276</ByteRangeEnd>
  <IfMatch>"828ef3fdfa96f00ad9f27c383fc9ac7f"</IfMatch>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetObject>
```

Amazon S3 returns the following response and the remainder of the file.

```
<GetObjectResponse xmlns="http://s3.amazonaws.com/doc/2006-03-01">
  <GetObjectResponse>
    <Status>
      <Code>200</Code>
      <Description>OK</Description>
    </Status>
    <Metadata>
      <Name>Content-Type</Name>
      <Value>text/plain</Value>
    </Metadata>
    <Metadata>
      <Name>family</Name>
      <Value>>Muntz</Value>
    </Metadata>
    <Data>--remainder of bigfile--</Data>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
  </GetObjectResponse>
</GetObjectResponse>
```

Versioned GetObject

The following request returns the specified version of the object in the bucket.

```
<GetObject xmlns="http://doc.s3.amazonaws.com/2006-03-01">
<Bucket>quotes</Bucket>
<Key>Nelson</Key>
<GetMetadata>true</GetMetadata>
<GetData>true</GetData>
<InlineData>true</InlineData>
<AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
<Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
<Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetObject>
```

Sample Response

```
<GetObjectResponse xmlns="http://s3.amazonaws.com/doc/2006-03-01">
<GetObjectResponse>
<Status>
<Code>200</Code>
```

```
<Description>OK</Description>
</Status>
<Metadata>
<Name>Content-Type</Name>
<Value>text/plain</Value>
</Metadata>
<Metadata>
<Name>family</Name>
<Value>Muntz</Value>
</Metadata>
<Data>aGEtaGE=</Data>
<LastModified>2006-01-01T12:00:00.000Z</LastModified>
<ETag>&amp;quot;828ef3fdfa96f00ad9f27c383fc9ac7f&amp;quot;</ETag>
</GetObjectResponse>
</GetObjectResponse>
```

REST GET Error Recovery

If an object GET fails, you can get the rest of the file by specifying the range to download. To do so, you must get the size of the object using `ListBucket` and perform a range `GET` on the remainder of the file. For more information, see GetObjectExtended (p. 347).

Related Resources

Operations on Objects (p. 334)

GetObjectExtended

> **Note**
> SOAP support over HTTP is deprecated, but it is still available over HTTPS. New Amazon S3 features will not be supported for SOAP. We recommend that you use either the REST API or the AWS SDKs.

`GetObjectExtended` is exactly like GetObject (p. 342), except that it supports the following additional elements that can be used to accomplish much of the same functionality provided by HTTP GET headers (go to http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html).

GetObjectExtended supports the following elements in addition to those supported by GetObject:

- *`ByteRangeStart, ByteRangeEnd:`* These elements specify that only a portion of the object data should be retrieved. They follow the behavior of the HTTP byte ranges (go to http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.35).
- *`IfModifiedSince:`* Return the object only if the object's timestamp is later than the specified timestamp. (http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.25)
- *`IfUnmodifiedSince:`* Return the object only if the object's timestamp is earlier than or equal to the specified timestamp. (go to http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.28)
- *`IfMatch:`* Return the object only if its ETag matches the supplied tag(s). (go to http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.24)
- *`IfNoneMatch:`* Return the object only if its ETag does not match the supplied tag(s). (go to http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.26)
- *`ReturnCompleteObjectOnConditionFailure:`* ReturnCompleteObjectOnConditionFailure: If true, then if the request includes a range element and one or both of IfUnmodifiedSince/IfMatch elements, and the condition fails, return the entire object rather than a fault. This enables the If-Range functionality (go to http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.27).

DeleteObject

Note
SOAP support over HTTP is deprecated, but it is still available over HTTPS. New Amazon S3 features will not be supported for SOAP. We recommend that you use either the REST API or the AWS SDKs.

The `DeleteObject` operation removes the specified object from Amazon S3. Once deleted, there is no method to restore or undelete an object.

Note
If you delete an object that does not exist, Amazon S3 will return a success (not an error message).

Example

This example deletes the "Nelson" object from the "quotes" bucket.

Sample Request

```
<DeleteObject xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <AWSAccessKeyId> AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</DeleteObject>
```

Sample Response

```
<DeleteObjectResponse xmlns="http://s3.amazonaws.com/doc/2006-03-01">
  <DeleteObjectResponse>
    <Code>200</Code>
    <Description>OK</Description>
  </DeleteObjectResponse>
</DeleteObjectResponse>
```

Elements

- *`Bucket:`* The bucket that holds the object.
- *`Key:`* The key that identifies the object.

Access Control

You can delete an object only if you have `WRITE` access to the bucket, regardless of who owns the object or what rights are granted to it.

GetObjectAccessControlPolicy

Note
SOAP support over HTTP is deprecated, but it is still available over HTTPS. New Amazon S3 features will not be supported for SOAP. We recommend that you use either the REST API or the AWS SDKs.

The `GetObjectAccessControlPolicy` operation fetches the access control policy for an object.

Example

This example retrieves the access control policy for the "Nelson" object from the "quotes" bucket.

Sample Request

```
<GetObjectAccessControlPolicy xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetObjectAccessControlPolicy>
```

Sample Response

```
<AccessControlPolicy>
  <Owner>
    <ID>a9a7b886d6fd24a541bf9b1c61be666e9</ID>
    <DisplayName>chriscustomer</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>a9a7b841bf9b1c61be666e9</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/global/AllUsers<URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

Response Body

The response contains the access control policy for the bucket. For an explanation of this response, SOAP Access Policy .

Access Control

You must have `READ_ACP` rights to the object in order to retrieve the access control policy for an object.

SetObjectAccessControlPolicy

Note
SOAP support over HTTP is deprecated, but it is still available over HTTPS. New Amazon S3 features will not be supported for SOAP. We recommend that you use either the REST API or the AWS SDKs.

The `SetObjectAccessControlPolicy` operation sets the access control policy for an existing object. If successful, the previous access control policy for the object is entirely replaced with the specified access control policy.

Example

This example gives the specified user (usually the owner) `FULL_CONTROL` access to the "Nelson" object from the "quotes" bucket.

Sample Request

```
<SetObjectAccessControlPolicy xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
       <ID>a9a7b886d6fd24a52fe8ca5bef65f89a64e0193f23000e241bf9b1c61be666e9</ID>

        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</SetObjectAccessControlPolicy>
```

Sample Response

```
<SetObjectAccessControlPolicyResponse xmlns="http://s3.amazonaws.com/doc/2006-
03-01">
  <SetObjectAccessControlPolicyResponse>
    <Code>200</Code>
    <Description>OK</Description>
  </SetObjectAccessControlPolicyResponse>
</SetObjectAccessControlPolicyResponse>
```

Access Control

You must have `WRITE_ACP` rights to the object in order to set the access control policy for a bucket.

SOAP Error Responses

Note
SOAP support over HTTP is deprecated, but it is still available over HTTPS. New Amazon S3 features will not be supported for SOAP. We recommend that you use either the REST API or the AWS SDKs.

In SOAP, an error result is returned to the client as a SOAP fault, with the HTTP response code 500. If you do not receive a SOAP fault, then your request was successful. The Amazon S3 SOAP fault code is comprised of a standard SOAP 1.1 fault code (either "Server" or "Client") concatenated with the Amazon S3-specific error code. For example: "Server.InternalError" or "Client.NoSuchBucket". The SOAP fault

string element contains a generic, human readable error message in English. Finally, the SOAP fault detail element contains miscellaneous information relevant to the error.

For example, if you attempt to delete the object "Fred", which does not exist, the body of the SOAP response contains a "NoSuchKey" SOAP fault.

The following example shows a sample SOAP error response.

```
<soapenv:Body>
  <soapenv:Fault>
    <Faultcode>soapenv:Client.NoSuchKey</Faultcode>
    <Faultstring>The specified key does not exist.</Faultstring>
    <Detail>
      <Key>Fred</Key>
    </Detail>
  </soapenv:Fault>
  </soapenv:Body>
```

The following table explains the SOAP error response elements

Name	Description
Detail	Container for the key involved in the error Type: Container Ancestor: Body.Fault
Fault	Container for error information. Type: Container Ancestor: Body
Faultcode	The fault code is a string that uniquely identifies an error condition. It is meant to be read and understood by programs that detect and handle errors by type. For more information, see List of Error Codes (p. 3). Type: String Ancestor: Body.Fault
Faultstring	The fault string contains a generic description of the error condition in English. It is intended for a human audience. Simple programs display the message directly to the end user if they encounter an error condition they don't know how or don't care to handle. Sophisticated programs with more exhaustive error handling and proper internationalization are more likely to ignore the fault string. Type: String Ancestor: Body.Fault
Key	Identifies the key involved in the error Type: String Ancestor: Body.Fault

Glossary

100-continue	A method that enables a client to see if a server can accept a request before actually sending it. For large `PUT`s, this can save both time and bandwidth charges.
account	AWS account associated with a particular developer.
authentication	The process of proving your identity to the system.
bucket	A container for objects stored in Amazon S3. Every object is contained within a bucket. For example, if the object named `photos/puppy.jpg` is stored in the `johnsmith` bucket, then it is addressable using the URL `http://johnsmith.s3.amazonaws.com/photos/puppy.jpg`
canned access policy	A standard access control policy that you can apply to a bucket or object. Valid Values: `private | public-read | public-read-write | authenticated-read | bucket-owner-read | bucket-owner-full-control`
canonicalization	The process of converting data into a standard format that will be recognized by a service such as Amazon S3.
consistency model	The method through which Amazon S3 achieves high availability, which involves replicating data across multiple servers within Amazon's data centers. After a "success" is returned, your data is safely stored. However, information about the changes might not immediately replicate across Amazon S3.
key	The unique identifier for an object within a bucket. Every object in a bucket has exactly one key. Since a bucket and key together uniquely identify each object, Amazon S3 can be thought of as a basic data map between "bucket + key" and the object itself. Every object in Amazon S3 can be uniquely addressed through the combination of the web service endpoint, bucket name, and key, as in http://doc.s3.amazonaws.com/2006-03-01/AmazonS3.wsdl, where "doc" is the name of the bucket, and "2006-03-01/AmazonS3.wsdl" is the key.
metadata	The metadata is a set of name-value pairs that describe the object. These include default metadata such as the date last modified and standard HTTP metadata such as Content-Type. The developer can also specify custom metadata at the time the Object is stored.
object	The fundamental entities stored in Amazon S3. Objects consist of object data and metadata. The data portion is opaque to Amazon S3.
part	The fundamental entities stored in Amazon S3. Objects consist of object data and metadata. The data portion is opaque to Amazon S3.

service endpoint	The host and port with which you are trying to communicate within the destination URL. For virtual hosted-style requests, this is `mybucket.s3.amazonaws.com`. For path-style requests, this is `s3.amazonaws.com`